

An Introduction to Modeling Structure from Sequence

UNIT 5.1

There are literally millions of protein sequences in the various sequence databases, but there are only a few tens of thousands of protein structures in the Protein Data Bank. The rate of growth of new sequences is a steeply rising exponential curve; that of new structures, if exponential at all, is much shallower. There is no possibility that the number of structures will ever approach, much less equal, the number of sequences. So what is the point of initiatives such as Structural Genomics? What sense does it make to be the tortoise in a race in which the hare has already won?

The underlying premise behind all attempts to determine a large number of diverse structures is that the total number of protein domain folds is much smaller, by many orders of magnitude, than the total number of sequences; in other words, that many sequences adopt essentially the same fold. If the fold of a protein could be recognized from sequence information alone, then a complete database of all possible folds would allow the structure corresponding to any sequence to be modeled, to at least some level of accuracy.

How reasonable is this assumption? It depends first of all on the reality of the limited universe of domain folds. (For the purpose of this discussion, the term “domain” means any part of the structure of a protein that is sufficiently compact so as to give the impression that it could fold stably without the rest of the protein. Although there are various mathematical/topological definitions of a domain, most domains are like Supreme Court Justice Potter Stewart’s 1964 explanation of pornography: we may not know how to define it, but we usually know it when we see it.) The best evidence that this universe is indeed limited is the diminishing number of new folds found every year despite the sharp increase in new structures (Hou et al., 2005). Simple application of Fisher statistics to this frequency distribution gives a crude estimate of the total number of folds. A recent attempt at cataloging estimates this number to be around 4000, of which nearly half (1700) are already known (Sadreyev and Grishin, 2006).

Therefore, there is reason to assume that the total number of folds will be known eventually and that it will indeed be many orders of magnitude less than the number of sequences. The problem of assigning a fold for every sequence now reduces to two steps: identifying the fold that corresponds to a given sequence, and deriving the best possible atomic model for that the structure of that sequence given knowledge of its domain fold(s).

That doesn’t sound so difficult, but in practice it has proven to be a formidable challenge. Both steps are far from straightforward in all but the simplest cases, and both represent very active areas of investigation. It is these steps that are the subjects of the protocols in this chapter.

We begin with a discussion of what would seem to be the easiest situation: homology modeling of a protein structure from a sequence that displays significant identity to one adopting a known fold. This is the subject of *UNIT 5.6* by Andrej Sali and colleagues, who have made some of the most important contributions to homology modeling. They discuss every aspect of the procedure, from fold assignment to alignment of the target with the template to model construction and validation. They emphasize that even very similar sequences may have regions of structure that diverge significantly (principally loops).

Modeling Structure from Sequence

5.1.1

Contributed by Gregory A. Petsko

Current Protocols in Bioinformatics (2006) 5.1.1-5.1.3

Copyright © 2006 by John Wiley & Sons, Inc.

Supplement 15

They show how multiple sequence alignments and the use of a family of templates can improve the accuracy of such regions. They also explain how to decide what size grain of salt should be used in taking the results of a homology model as factual. Their program, MODELLER, is one of the most widely used tools for homology model construction, and they describe in detail how to use it.

A different approach to model construction is discussed in the unit by Umeyama and Iwate. Their program FAMS (*UNIT 5.2*) uses a simulated annealing algorithm to “refine” the model so as to improve the accuracy, particularly of the soft variables (torsion angles). Since this program is fully automated, it has some appeal for less sophisticated users who may not be willing or able to try different strategies to obtain a suitable model.

I have always believed that, although integral membrane protein structures are the most difficult type to determine experimentally, they ought to be among the easiest to model. In general, their topologies are much simpler than those of soluble proteins: for example, mixed α -helical and β -sheet domains in the membrane are essentially unknown. Membrane-spanning domains tend to be either bundles of α -helices or barrels of antiparallel β strands, both of which are relatively easy to recognize in amino acid sequences. Although the available database of membrane protein structures is still quite limited, enough patterns have already begun to emerge to give confidence that this type of modeling will eventually become common. Considering that over half of all known drugs target integral membrane proteins, mostly G-protein coupled receptors and ion channels, it is also likely that such modeling will have considerable practical importance.

In the third unit in this chapter, a collaborative team from the Hebrew University in Jerusalem and the Lawrence Berkeley Laboratory in California describes a tool for predicting the structures of simple α -helical bundle membrane proteins (*UNIT 5.3*). By running a global molecular dynamics search of configuration space, the protocol generates a set of candidate structures. The best one is selected from among these using the silent amino acid substitutions in the protein family as a stringent test for robustness. It seems likely that this procedure is just the tip of the proverbial iceberg for membrane protein prediction.

Homology modeling demands that the model be inspected, not only by computer program but also by eye. For this and numerous other reasons, the ability to display and manipulate the three-dimensional structures of proteins has passed from the province of a select few into the routine toolkit of almost every biologist. Among the many public software packages available for this purpose, RasMol (*UNIT 5.4*) is one of the oldest, most versatile, and easiest to use. In *UNIT 5.4*, David Goodsell gives an overview of its capabilities and then describes a number of useful protocols that should not only familiarize readers with RasMol, but also enable them to carry out many of the most common procedures.

New units in this chapter address two other important issues in structure modeling. One of the most frequently asked questions about any “new” protein structure is: does it resemble any previously known fold? This is not just an academic matter. Increasingly, protein structures are being determined for gene products of unknown function, not only because of the structural genomics initiatives but also because genetics often leads to the identification of a sequence as being an important contributor to, for example, a human disease, but there is no information from sequence comparisons about what the biochemical or biological function(s) of the gene product might be. The hope is that, since structure changes much more slowly than sequence, similarity to a structure of known function might provide a valuable clue.

I am not completely sanguine about this belief. On the one hand, there are some impressive examples of its success (Kim et al., 2004). On the other hand, it is clear that the coupling

between overall fold and biochemical function is often quite loose, especially for some protein superfamilies (Hegyi and Gerstein, 2001). Nevertheless, comparing a protein's fold with those already known is an important and sometimes powerful method. Liisa Holm, whose program DALI (UNIT 5.5) is the most widely used tool for this purpose, describes in her unit in this chapter how that tool should be employed. As the pace of structure determination increases, DALI will be in the vanguard not only for comparison of structures but also for assembling the database of fold libraries and assessing fold divergence.

The growth of structure determination has turned most biochemists and biologists into consumers of structural information. Genomics is accelerating this trend. As the demand for such information continues to outstrip the supply, all aspects of structure modeling assume increasing importance. For those who have yet to try their hand at such endeavors, the encouraging news is that the tools are getting easier to use as well as more accurate. Dip into the protocols in this chapter and see!

LITERATURE CITED

- Hegyi, H. and Gerstein, M. 2001. Annotation transfer for genomics: Measuring functional divergence in multi-domain proteins. *Genome Res.* 11:1632-1640.
- Hou, J., Jun, S.R., Zhang, C., and Kim, S.H. 2005. Global mapping of the protein structure space and application in structure-based inference of protein function. *Proc. Natl. Acad. Sci. U.S.A.* 102:3651-3656.
- Kim, Y., Yakunin, A.F., Kuznetsova, E., Xu, X., Pennycooke, M., Gu, J., Cheung, F., Proudfoot, M., Arrowsmith, C.H., Joachimiak, A., Edwards, A.M., and Christendat, D. 2004. Structure- and function-based characterization of a new phosphoglycolate phosphatase from *Thermoplasma acidophilum*. *J. Biol. Chem.* 279:517-526.
- Sadreyev, R.I. and Grishin, N.V. 2006. Exploring dynamics of protein structure determination and homology-based prediction to estimate the number of superfamilies and folds. *BMC Struct. Biol.* 6:6.

Contributed by Gregory A. Petsko
Brandeis University
Waltham, Massachusetts

The computer program FAMS (Full Automatic Modeling System; Ogata et al., 2000; Iwadata et al. 2001) performs homology modeling of protein structures by means of an algorithm consisting of database searches and simulated annealing. FAMS produces a model in which the torsion angles of the backbone and sidechains are highly accurate.

An overview of the processes for obtaining a protein model via FAMS is shown in Figure 5.2.1. This unit describes a procedure for searching FAMSBASE (Yamaguchi et al., 2003), the database of structural models calculated by FAMS (see Basic Protocol).

CHECKING FAMSBASE FOR A PROTEIN MODEL

When a 3-D structural model is required for a particular protein, one should first check whether or not the protein is already modeled. FAMSBASE is a relational database of comparative protein structure models for the entire genomes of 41 species, as presented in the GTOP (Genomes TO Protein structures and functions) database at <http://spock.genes.nig.ac.jp/~gtop-old/gtop.html>. The models in that database were all calculated using FAMS. FAMSBASE provides versatile search and query functions, including searching by name of ORF (open reading frame), ORF annotation, Protein Data Bank (PDB) ID, and sequence similarity. FAMSBASE is available online at <http://famsbase.bio.nagoya-u.ac.jp/famsbase/>. The present percentage of ORFs with 3-D protein models in FAMSBASE is 42%; therefore, requested protein models are currently available in approximately half of all cases.

Necessary Resources

Hardware

Any computer with an Internet connection

Software

Web browser (Internet Explorer v. 5.0 or later or Netscape v. 4.7 or later for Windows; Internet Explorer v. 4.5 or later for Macintosh)

1. Log in to FAMSBASE as follows.

- a. Go to the URL of FAMSBASE, <http://famsbase.bio.nagoya-u.ac.jp/famsbase/>.

Figure 5.2.2 shows the login page of FAMSBASE.

- b. Enter a login name and password. If accessing the database for the first time, obtain a login name and a password by clicking the link labeled “For the first user.” Alternatively, click on the “Public Login” hyperlink.

After logging in, one arrives at the FAMSBASE search page. Figure 5.2.3 shows the upper part of the search page; Figure 5.2.4 shows the lower part. “Public Login” only provides sufficient access to determine whether or not a model exists in FAMSBASE. Individuals who select “Public Login” cannot view structures.

2. Specify search criteria.

- a. *Species.* The upper part of the search page (Fig. 5.2.3; Section 1) lists 41 species whose genome ORFs have been determined. The check boxes on the left-hand side of the query form allow the user to specify which species should be included

BASIC PROTOCOL

Modeling Structure from Sequence

5.2.1

in the search. It is possible to select multiple species. Figure 5.2.5 shows an example in which *Escherichia coli* is selected.

More details on the 41 species are described in the GTOP homepage, <http://spock.genes.nig.ac.jp/~gtop-old/org.html>, which contains the results not only of PSI-BLAST but also of FASTA and normal BLAST, among others (Pearson and Lipman, 1988; Altschul et al., 1990).

- b. The lower part of the search page provides the following text boxes and radio buttons for searching: (2) Search for ORFs by Gene (ORF) Name; (3) Search for ORFs by PDB ID of Reference Protein; (4) Search for ORFs by Motif Name; and (5) Search for ORFs by FAMS Results.

The gene name used in the Search for ORFs by Gene (ORF) Name text box is based on the gene names used in the GTOP Web site mentioned above. The motifname used in the Search for ORFs by Motif Name text box is based on the PROSITE motifs, <http://us.expasy.org/prosite/>. The FAMS results used in Search for ORFs by FAMS Results means whether or not the model exists in the database. As an example, Figure 5.2.6 shows a query for Gene Name, "abc."

Once the search criteria have been entered, click the Search button at the top of the Web page.

- c. Alternatively, there are two additional text boxes in the lower part of the search page: Search for ORFs by Hetero Atom of Reference Protein and Search for ORFs by Amino Acid Sequence. After entering the corresponding information in the text box(es), click the Search button (for Search for ORFs by Hetero Atom of Reference Protein) or the Submit Query button (for Search for ORFs by Amino Acid Sequence).

In the Search for ORFs by Hetero Atom of Reference Protein text box, the Hetero Atom refers to the HETATM line in PDB format. An amino acid sequence search using FASTA (UNIT 3.9) is performed by the Search for ORFs by Amino Acid Sequence text box (Fig. 5.2.7). Users can search by several criteria at once, but the Amino Acid Sequence search is exclusive.

Select a model

3. Examine the model list that appears (Fig. 5.2.8) with annotations of ORFs, model lengths (number of amino acid residues), and identity percentages of amino acid sequence alignments (with experimentally known structure).
4. Select one line in the model list by clicking on a template ID (in the PSIBlast column in Fig. 5.2.8) from the model list, which will then bring up the amino acid alignment view page (Fig. 5.2.9). Display the selected model structure by clicking on the View Target button.

Both the model and the template will be displayed simultaneously (Fig. 5.2.10) by clicking the Superimpose button when using an appropriate model viewer, e.g., RasMol, <http://www.umass.edu/microbio/rasmol/>. The model file (not containing the template) can also be downloaded by clicking on the View Target button (Fig. 5.2.11).

GUIDELINES FOR UNDERSTANDING RESULTS

Once the required model has been obtained, whether from FAMSBASE or from FAMS, one may wonder about its accuracy. Generally, if the query sequence and the amino acid sequence of the experimentally known structure share a high percent identity, this strongly supports the accuracy of the model structure. Quantitatively, if the percentage is >30%, the RMSD (root mean square distance) values are within ~4

Å (over the C_α backbone) of the true structure. Note that in low-homology cases, regions of locally high homology exist that may contain important information in a model. In cases of low percent identity (<30%), statistically half of all models whose alignment *E*-values are low enough (<10⁻³) will have a small enough RMSD (within 4 Å), to be considered accurate models. The *E*-value guarantees the length of the model. In the case of alignments of low-enough *E*-value, the reliable region is sufficiently large in comparison to the entire ORF region. After a few years, the number of high-identity-percentage models will increase, and, at that time, the homology-modeling method will produce more accurate protein structures.

COMMENTARY

Background Information

The authors of this unit developed a computer program, FAMS (Full Automatic Modeling System) to build model structures based on reference structures solved using X-ray diffraction, NMR, or other experimental methods, as well as amino acid sequence alignment between a target and its reference structure. FAMSBASE is a relational database of comparative protein structure models in GTOP (Genomes TO Protein structures and functions) alignment, calculated by FAMS. Both GTOP and FAMSBASE are projects of the Japanese government.

The basic FAMS algorithm consists of a database search and simulated annealing. The first step obtains the C_α coordinates, the second step, the backbone, the third step, side chains, and the last step, all atoms.

The effectiveness of the software was highlighted by its performance in the CAFASP2 and CAFASP3 competitions (Fischer et al., 2001), especially in terms of side-chain accuracy, with good performance in regard to the backbone as well. CAFASP (Critical Assessment of Fully Automated Structure Prediction) is a competition for determining the best software of this kind. Another competition, CASP (Critical Assessment of Techniques for Protein Structure Prediction) determines the best researcher in this area. CASP experiments were started in 1994 as CASP1, and continued biennially through to 2002 as CASP5. CAFASP experiments were started at the same time as CASP3, beginning with CAFASP1, and hence CAFASP3 was running in 2002. Results from the comparative modeling section of CASP5 suggested that fully automated building procedures were less accurate than procedures with human intervention (Iwata et al., 2001). Human intervention worked effectively on CASP5 and the assessments have highlighted

the algorithmic improvement of sequence alignments. However, fully-automated procedures are essential, and, indeed, have been used for large-scale genome modeling. CAFASP3 assessments did not judge human intervention, but only software performance.

The use of typical alignment software such as FASTA (UNIT 3.9), BLAST (UNITS 3.3 & 3.4), or PSI-BLAST to determine which modeling software demonstrates the best performance is very important, and the results are of interest not only to computational biologists but also to biologists at the laboratory bench.

Suggestions for Further Analysis

It is currently not possible to access the FAMS server. However, the authors expect that in the future, researchers will be able to submit novel sequences directly to FAMS in order to obtain structure predictions (see Fig. 5.2.12 for the FAMS Web page).

Literature Cited

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215:403-410.
- Fischer, D., Elofsson, A., Rychlewski, L., Pazos, F., Valencia, A., Rost, B., Ortiz, A.R., and Dunbrack, R.L., Jr. 2001. CAFASP2: The second critical assessment of fully automated structure prediction methods. *Proteins* 45:171-183.
- Iwata, M., Ebisawa, K., and Umeyama, H. 2001. Comparative modeling of CAFASP2 competition. *Chem-Bio. Informatics J.* 1:136-148.
- Ogata, K. and Umeyama, H. 2000. An automatic homology modeling method consisting of database searches and simulated annealing. *J. Mol. Graph. Model.* 18:258-272, 305-256.
- Pearson, W.R. and Lipman, D.J. 1988. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. U.S.A.* 85:2444-2448.

Yamaguchi, A., Iwadate, M., Suzuki, E.-I., Yura, K., Kawakita, S., Umeyama, H., and Go, M. 2003. Enlarged FAMSBASE: Protein 3D structure models of genome sequences for 41 species. *Nucleic Acids Res.* 31:1-6.

<http://spock.genes.nig.ac.jp/~genome/gtop.html>

GTOP Web site.

Internet Resources

<http://physchem.pharm.kitasato-u.ac.jp/FAMS/>
FAMS Web site.

<http://famsbase.bio.nagoya-u.ac.jp/famsbase/>
FAMSBASE Web site.

Contributed by Hideaki Umeyama and
Mitsuo Iwadate
Kitasato University
Tokyo, Japan

Figures 5.2.1-5.2.12 appear on the following pages.

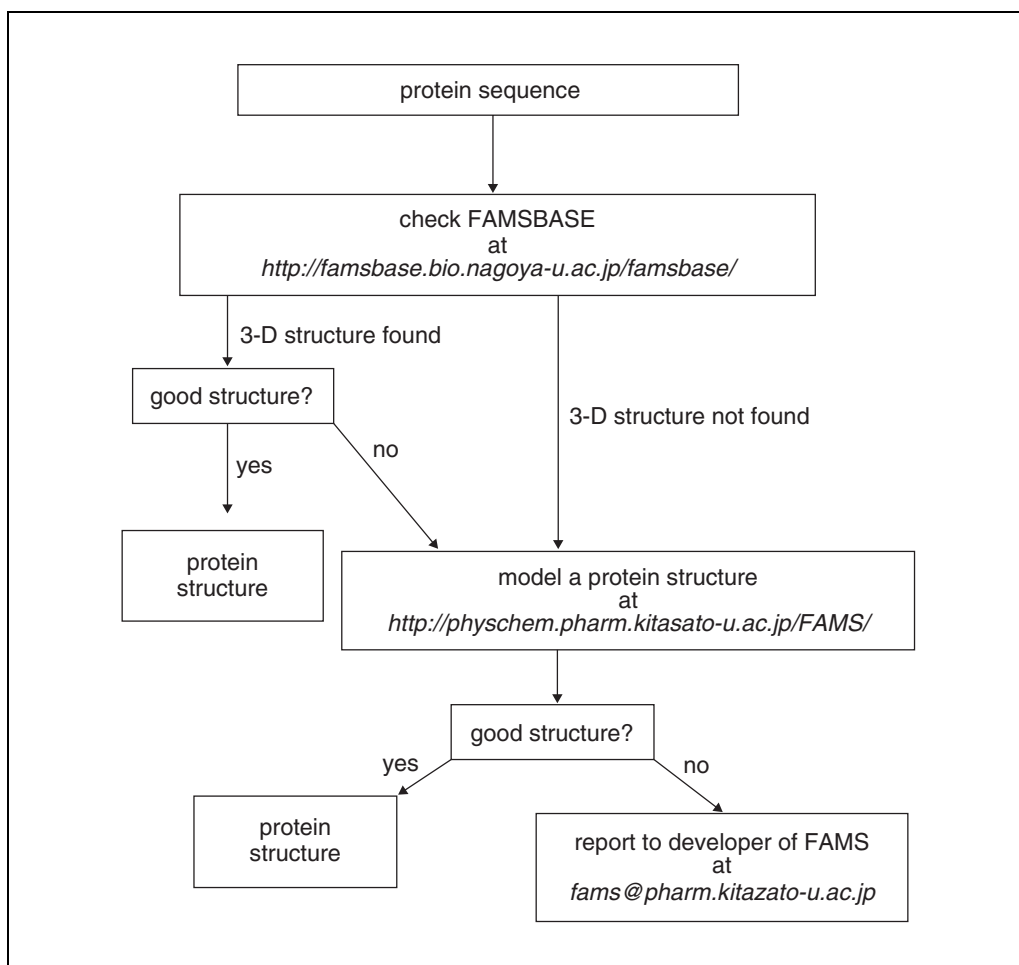


Figure 5.2.1 Flowchart of modeling by FAMS, from sequence to structure. Basic Protocol outlines the searching of FAMSBASE.

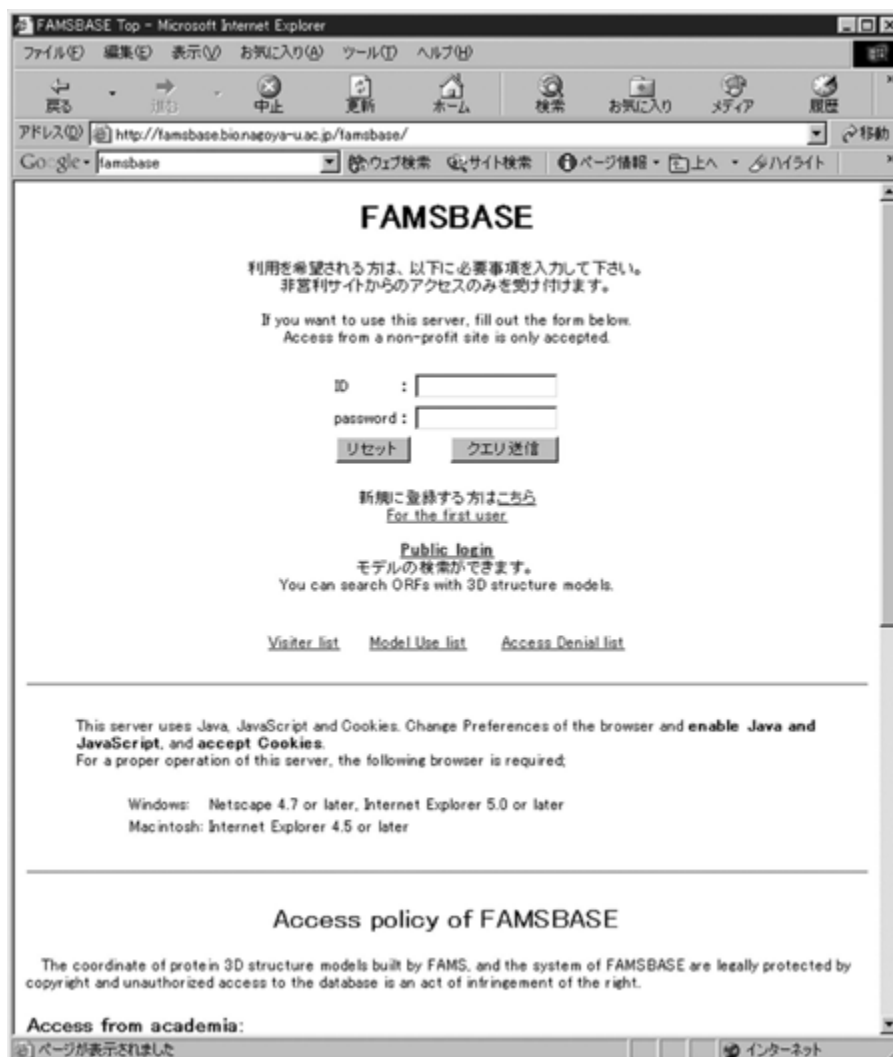


Figure 5.2.2 The login page of FAMSBASE. As stated on the page, one must first obtain an ID and password from an administrator of FAMSBASE. If time is a factor or one just wishes to check the contents of the database, click on the “Public login” link to go to the search page.

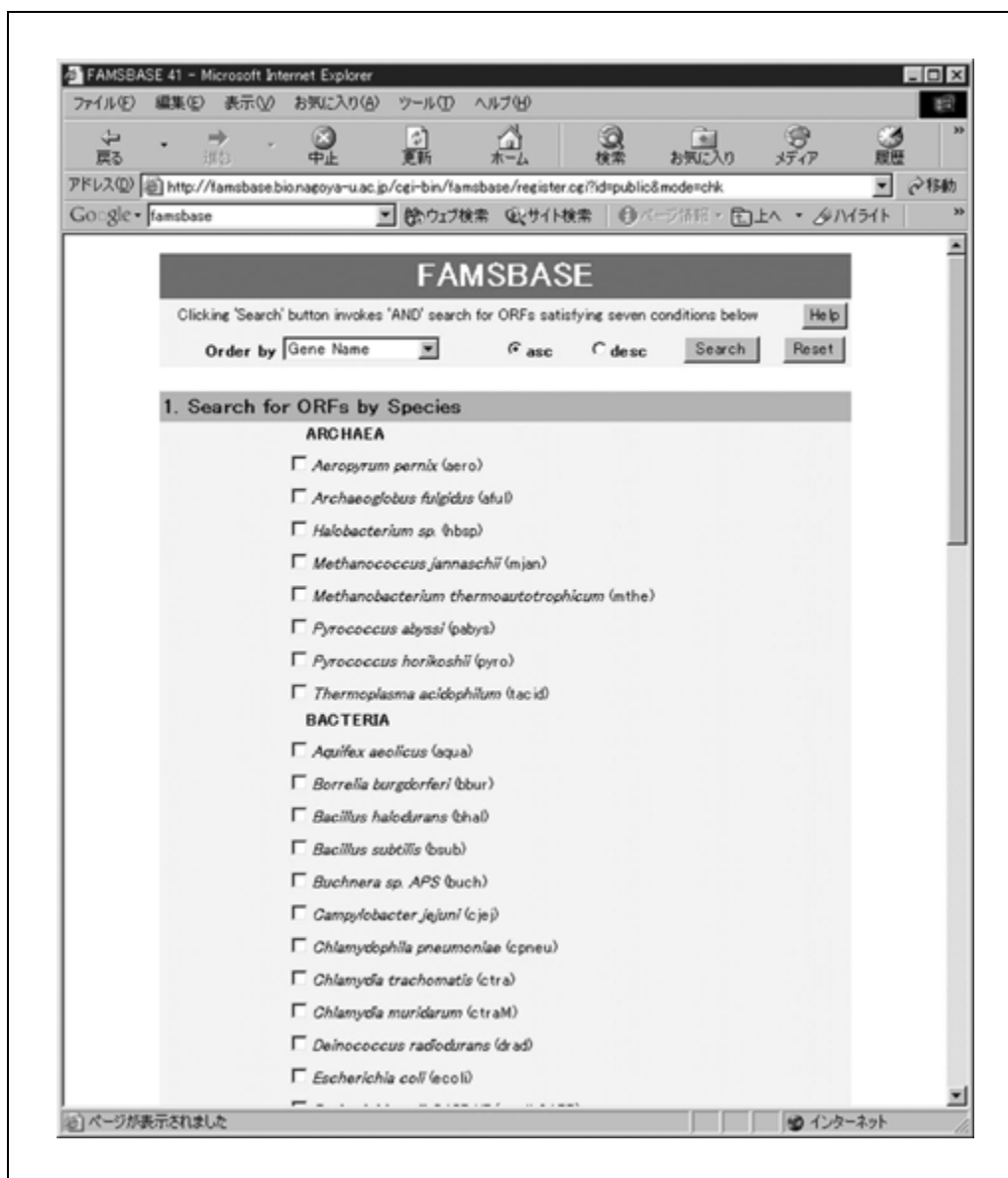


Figure 5.2.3 The upper part of the search page of FAMSBASE. 41 species whose genome ORFs have been determined are listed with check boxes on the left-hand side. More details of the 41 species are described in <http://spock.genes.nig.ac.jp/~gtop-old/org.html>.

FAMSBASE 41 - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

戻る 進む 中止 更新 ホーム 検索 お気に入り メディア

アドレス(AD) http://famsbase.bio.nagoya-u.ac.jp/cgi-bin/famsbase/register.cgi?id=public&mode=chk

Go: Google famsbase ウェブ検索 サイト検索 ページ情報 上へ ハイライト

VIRUSES

☐ Bacteriophage T4 (14)

2. Search for ORFs by Gene(ORF) Name

Gene Name: (ex.s10002)

3. Search for ORFs by PDB ID of Reference Protein

PDB ID: (ex.1RKD1EYH_A)

4. Search for ORFs by Motif Name

Keywords: (ex.amino)

5. Search for ORFs by Keywords in ORF Product

Keywords: (ex.RNA factor)

6. Search for ORFs by FAMS Results

☐ Search for ORFs with 3D structures produced by FAMS.

☐ Search for ORFs without 3D structures produced by FAMS.

☒ both

Search for ORFs by Hetero Atom of Reference Protein

Hetero Atom Code: (Hetero Atom List)

Search Reset

Search for ORFs by Amino Acid sequence

File Upload: 参照...

Or Enter query sequence:

Subset range: Submit Query Clear

Public login: 000457 Registered login: 000126 (since 2002.03.02)

ページが表示されました インターネット

Figure 5.2.4 The lower part of the search page of FAMSBASE. Text boxes and radio buttons for searching the database are provided.

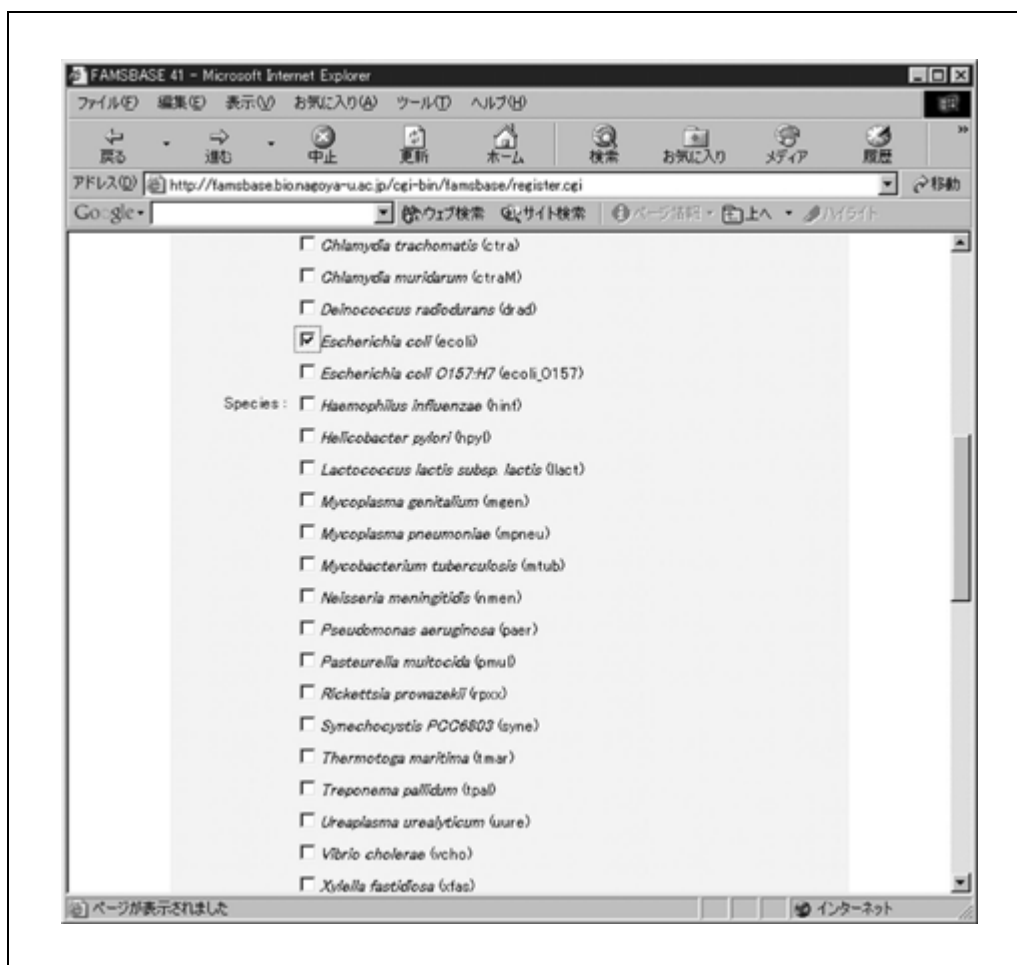


Figure 5.2.5 If a particular species is of interest, one may click the check boxes to the left of the species names. In this figure, *Escherichia coli* is selected.

FAMSBASE 41 - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

戻る 進む 中止 更新 ホーム 検索 お気に入りに メディア 履歴

アドレス(AD) http://famsbase.bionagoya-u.ac.jp/cgi-bin/famsbase/register.cgi 移動

Go:gle

2. Search for ORFs by Gene(ORF) Name

Gene Name: (ex:slf0002)

3. Search for ORFs by PDB ID of Reference Protein

PDB ID: (ex:1RKD 1EYH_A)

4. Search for ORFs by Motif Name

Keywords: (ex:amino)

5. Search for ORFs by Keywords in ORF Product

Keywords: (ex:RNA factor)

6. Search for ORFs by FAMS Results

☐ Search for ORFs **with** 3D structures produced by FAMS.

☐ Search for ORFs **without** 3D structures produced by FAMS.

☒ **both**

Search for ORFs by Hetero Atom of Reference Protein

Hetero Atom Code: (hetero Atom List)

Search Reset

Search for ORFs by Amino Acid sequence

File Upload: 参照...

Or Enter query sequence:

ページが表示されました インターネット

Figure 5.2.6 To search the database using an ORF or protein name, input the name directly into the text box. As an example, an ORF named “abc” has been input.

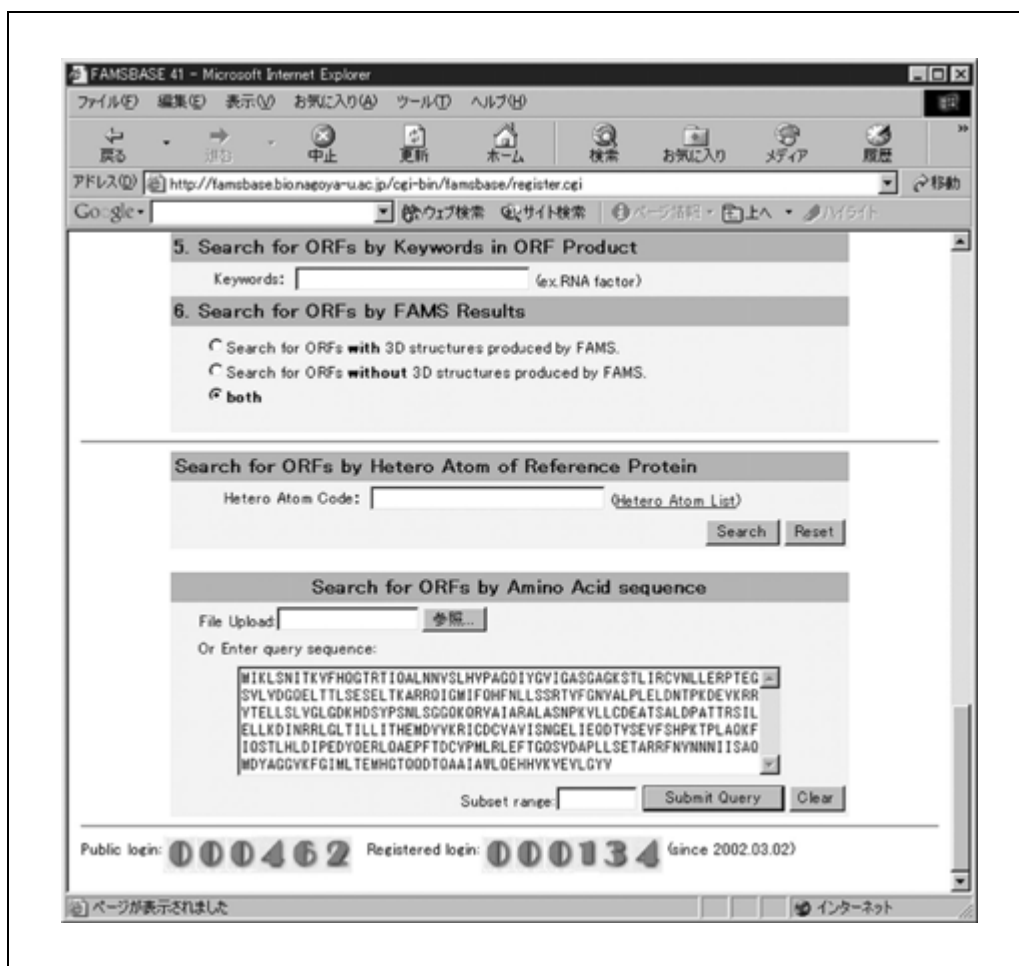


Figure 5.2.7 If an amino acid sequence is of interest, input the sequence in the large text box as shown here.

Searched ORFs

4 Hits Found Order by: Gene Name asc desc Search Help

Show: 100 Hits Goto No. 1 Goto Top Page Previous Page Next Page

No.	gene	genome	aa	product	PSIBlast	GTOP
1	abc_0	ecoli	343	"ATP-binding component of a transporter"	1G29_1	26.4 abc
2	abc_1	ecoli	343	"ATP-binding component of a transporter"	1B0U_A	32.4 abc
3	abc_2	ecoli	343	"ATP-binding component of a transporter"	1F2T_B	26.2 abc
4	abc_3	ecoli	343	"ATP-binding component of a transporter"	1E69_A	19.7 abc

Figure 5.2.8 A model list with annotations, model lengths (number of amino acids), and identity percentages of amino acid sequence alignments with experimentally known structure. To obtain a particular model, select one line by clicking on a template ID (shown in the PSIBlast column in this figure).

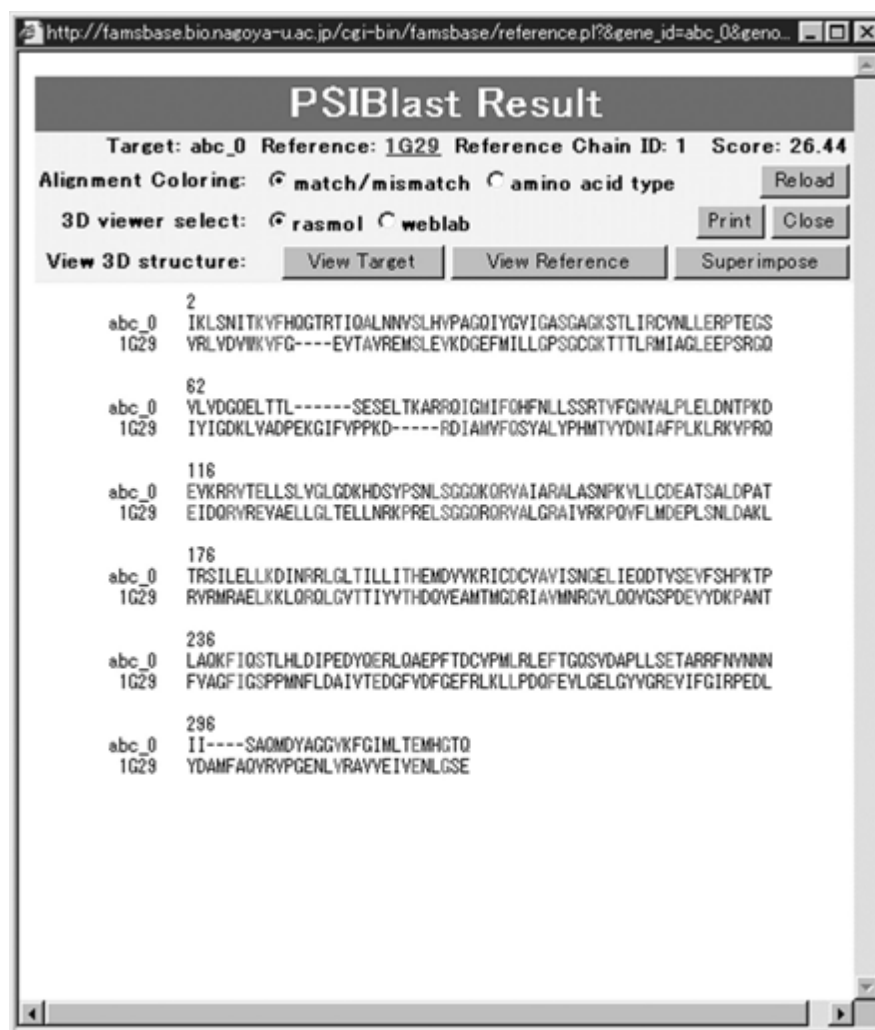


Figure 5.2.9 The amino acid alignment view page. To display the selected model, click the View Target button. Both the model and the template will be displayed by clicking the Superimpose button.

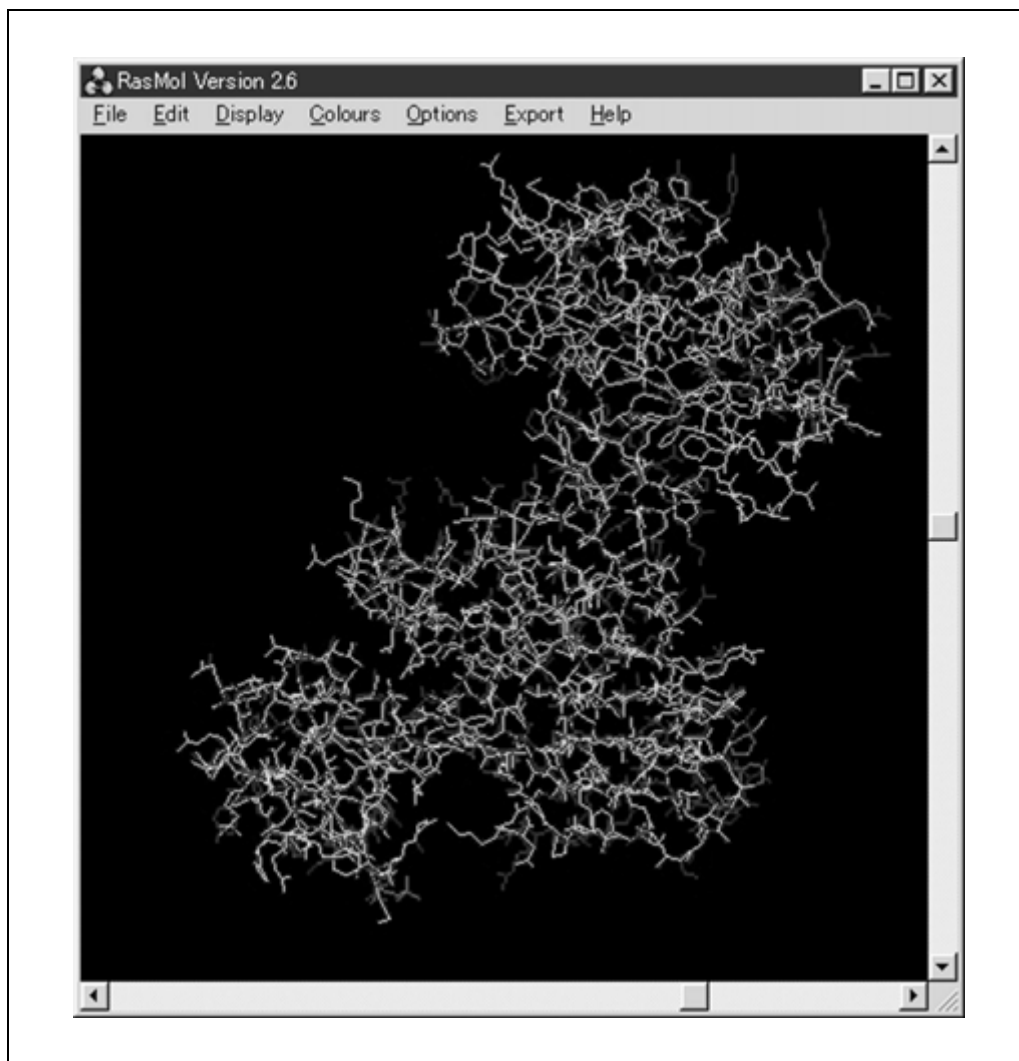


Figure 5.2.10 A Superimpose view using RasMol. The model is in blue and the template is in green. *This black-and-white facsimile of the figure is intended only as a placeholder; for full-color version of figure go to http://www.interscience.wiley.com/c_p/colorfigures.htm.*

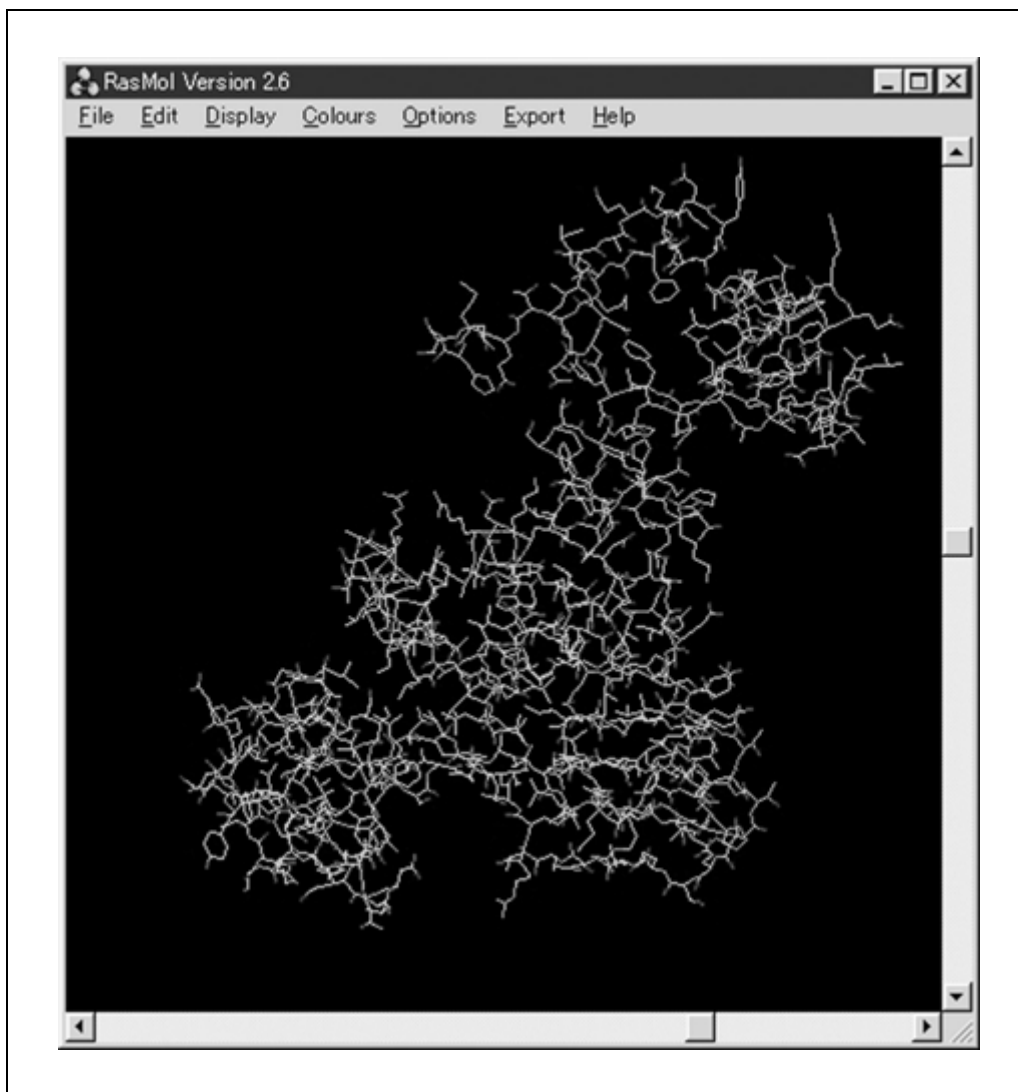


Figure 5.2.11 The model viewed after clicking the View Target button. *This black-and-white facsimile of the figure is intended only as a placeholder; for full-color version of figure go to http://www.interscience.wiley.com/c_p/colorfigures.htm.*



Figure 5.2.12 The FAMS Web page. The server status is displayed in the upper right-hand corner.

Modeling Membrane Proteins Utilizing Information from Silent Amino Acid Substitutions

Transmembrane α -helical bundles represent a simple topology that can be described by a relatively small number (n) of parameters: (1) helix tilt, (2) rotational position, and (3) register (Fig. 5.3.1). Thus for any hetero-oligomer, $3 \times n$ parameters are needed to describe the overall structure, while for any symmetrical homo-oligomer only 2 parameters are generally sufficient to describe the structure: helix tilt (β) and rotational pitch angle (ϕ).

Due to the reduced number of degrees of freedom, it is possible to exhaustively search each of the above parameters computationally in a procedure for which the name Global Molecular Dynamics Search (GMDS) has been coined (Adams et al., 1995). GMDS has been automated by a comprehensive series of task files and modules, written by Paul D. Adams, called CHI (CNS searching of Helix Interactions; Adams et al., 1995), to be used in the general computational structural biology software suite CNS (Crystallography and NMR System).

Depending on the parameters used, CHI routinely yields several candidate structures with a characteristic tilt and rotational pitch angle. Selection amongst the different candidate structures can be done using variety of procedures, such as the fitting of each structure to some experimental data (e.g., mutagenesis; Lemmon et al., 1992b; Treutlein et al., 1992; Arkin et al., 1994).

In this unit, a different procedure is described for the selection the correct structure from a list of plausible competing structures based on silent amino acid substitutions (see Basic Protocol). This procedure makes use of homology data in an objective manner to select the correct model, and, in principle, can be applied as a screening procedure whenever more than one model exists.

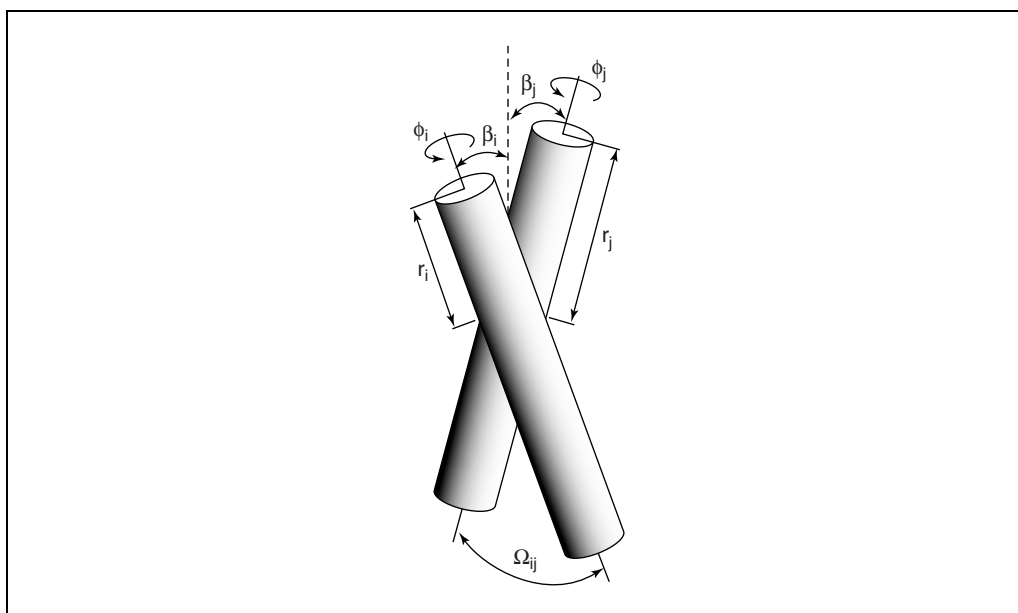


Figure 5.3.1 In a bundle with n transmembrane α -helices (helices i and j in this case), $3n$ parameters can be used to describe the general structure, assuming rigid helices: (1) the inclination of the helices with respect to the bundle axis, β_i , related to the commonly used crossing angle Ω ; (2) the rotational angle about the helix director, ϕ_i , which defines which side of helix i is facing towards the bundle core; and (3) the helix register, r_i , which defines the relative vertical position of the helix.

SELECTING A CORRECT PROTEIN STRUCTURE USING CHI

CHI is a series of user-friendly task files and modules written by Adams (1995) to be used in the general software suite CNS (Crystallography and NMR System; Brünger et al., 1998). CHI constructs multiple bundles of helices, each differing from the other by the rotation of the helices about their axes, as well as the bundle handedness. These are then used as starting positions for molecular dynamics simulations and energy minimization protocols. The output structures from these simulations are compared and grouped into clusters that contain similar structures. An average of the structures forming a cluster represents a model with characteristic interhelical interactions and helix tilt. The “Silent Amino Acid Substitution Protocol” performs the above simulations on close sequence variants that are likely to share the same structure, followed by a comparison of the clusters from the different variants, in an attempt to find a common cluster into which all these variants fold.

In the protocol it will be assumed that the user is using a generic Unix system employing the csh or tcsh shell. The commands are entered at a terminal with the %> command prompt. Text files are edited using a text editor. Those who are unfamiliar with the Unix environment should refer to *APPENDIX 1C* & *APPENDIX 1D*.

Necessary Resources

Hardware

Hardware requirements are defined by those that are officially supported by CNSsolve, i.e., one of the following computers:

- SGI (R4000 and later) running IRIX 4.0.5 or later
- HP (PA Risc) running HP-UX 9.05 or later
- DEC Alpha running OSF1/Digital Unix/Tru64 Unix
- PC (i386, i486, i586, or i686) running Linux or Windows 98 or NT or higher

Additionally, CNSsolve also provides unsupported installations for other systems:

- Convex running ConvexOS:
- Cray (J90, YMP, C90, T90) running Unicos
- Cray T3E (single CPU) running Unicasmk
- IBM RS/6000 running AIX
- Sun running SunOS
- Unix systems with g77/gcc (EGCS-1.1)
- Windows 98 or NT (or higher) systems with g77/gcc (EGCS-1.1)

A Macintosh OS X port is also available (contact the authors for details; arkin@cc.huji.ac.il)

Software

CNSsolve: available free of charge for academic users at <http://cns.csb.yale.edu>

CHI: available from Paul D. Adams (PDAdams@lbl.gov)

Perl: Perl is a component of nearly all standard Unix distributions. It is available free of charge at www.perl.org. Install according to the instructions on the Web page.

Three Perl scripts: (1) `ak_cluster.pl`, (2) `compare_rmsd.pl`, and (3) `to_gly.pl` (available from the authors; arkin@cc.huji.ac.il)

A CNSsolve input script, `cns.inp` (available from the authors; arkin@cc.huji.ac.il)

A standard text editor, e.g., jot, notepad, or nedit)

A Web browser

Software to perform multiple sequence alignment (e.g., ClustalX, ClustalW, or Pileup from the GCG Wisconsin package)

Install software and set up environment

1. Install CNSsolve as follows (more detailed installation instructions can be found on the CNSsolve Web page, <http://cns.csb.yale.edu>):

- a. Uncompress and extract the CNSsolve tar archive in /usr/local/:

```
%> tar -xzf cns_solve_1.1_basic_inputs.tar.gz
```

- b. Assuming that the above file was uncompressed in /usr/local/ there is now a new directory:

```
/usr/local/cns_solve_1.1/
```

- c. Using any text editor, edit the file:

```
/usr/local/cns_solve_1.1/cns_solve_env
```

by changing only one line, as follows (assuming that CNSsolve is located in /usr/local/cns_solve_1.1/):

```
setenv CNS_SOLVE /usr/local/cns_solve_1.1
```

- d. In order to compile the program, in the CNSsolve directory that was created in substep 1b (/usr/local/cns_solve_1.1/), type:

```
%> make install
```

This process may take several minutes depending on the computer platform, at the end of which there is a new executable program called cns.

2. Install CHI as follows:

- a. Uncompress and extract the CHI tar archive:

```
%> tar -xzf chi.tar.gz
```

- b. Assuming that the above file was uncompressed in /usr/local/, there is now a new directory:

```
/usr/local/chi/
```

- c. Using a text editor edit the file:

```
/usr/local/chi/chi_env
```

by changing only one line, as follows (assuming that CHI is located in /usr/local/chi):

```
setenv CHI_ROOT /usr/local/chi
```

- d. In order to compile the program, in the src directory of CHI (/usr/local/chi/src/) type:

```
%> make
```

3. Place the three Perl scripts (ak_cluster.pl, compare_rmsd.pl, and to_gly.pl) in /usr/local/chi/bin/.
4. Place the file cns.inp in /usr/local/chi/bin/.

5. In order for the system to recognize both CNS and CHI, which have been recently compiled, edit the `.cshrc` file (APPENDIX 1C) to include the following two lines:

```
source /usr/local/chi/chi_env
source /usr/local/cns_solve 1.1/cns_solve env
```

Define the sequences for the GMDS

There are two considerations that one must take into account. The first is the identity of the transmembrane segments to be simulated. The transmembrane α -helices must therefore be delineated from the rest of the protein. The second is, what are the homologous sequences to one's protein of interest?

6. Determine the transmembranal amino acids range, either by prior knowledge or by using programs predicting transmembranal domains: e.g., via the interactive programs TMHMM (<http://www.cbs.dtu.dk/services/TMHMM/>) or PSIPRED (<http://bioinf.cs.ucl.ac.uk/psipred/>).
7. Search protein databases (e.g., NCBI, PDB, or GeneBank, all accessible from the NCBI home page: <http://www.ncbi.nlm.nih.gov>) for homologous sequences using the transmembrane segments determined above.

The minimal identity between the sequences should be kept very high, in order to ensure that all changes are indeed "silent." The authors typically use sequences that are at least 75% identical.

8. Perform multiple sequence alignment (MSA) of the desired homologous sequences using MSA programs—e.g., ClustalX, ClustalW (UNIT 2.3), or PileUp (UNIT 3.6) from the GCG Wisconsin package.

No gaps should be allowed, i.e., the length of all homologous sequences should be identical. The results of the MSA will make it possible to select the exact sequences from the homologous proteins that correspond to the transmembrane domains of the protein of interest.

Set up an appropriate directory structure

Since GMDS produces a large number of files, it is best to work in an orderly and organized fashion. The authors therefore recommend the following directory setup.

9. Create a directory that will contain all the subdirectories and files used in the GMDS (it will be assumed that this directory is directly under the home directory, e.g., `~/MyProtein`). Create a specific subdirectory in that directory for each variant (e.g., `~/MyProtein/variantA`, `~/MyProtein/variantB`, ... `~/MyProtein/variantN`).

Prepare the instructions file `chi_param`

In order to run the GMDS using CHI, all that is needed is a single instructions file called `chi_param`, which, as its name suggests, contains the parameters needed for a CHI run. `chi_param` can be generated by a Web server in the CHI site (Fig. 5.3.2; <http://www.csb.yale.edu/userguides/datamanip/chi/html/chi.html>). One can obtain the file simply by contacting the authors and editing it manually with any text editor. `chi_param` contains exhaustive comments making the editing of the file self-explanatory.

To create a new parameter file from scratch

- 10a. In the CHI main menu on the left-hand side of the CHI home page (Fig. 5.3.2), click on "Create setup."
- 11a. In the first "Create setup" screen that appears (Fig. 5.3.3), type the desired molecule name.

For convenience the name of the molecule should be identical to the subdirectory name (e.g., `variantA`).



Figure 5.3.2 CHI main page.

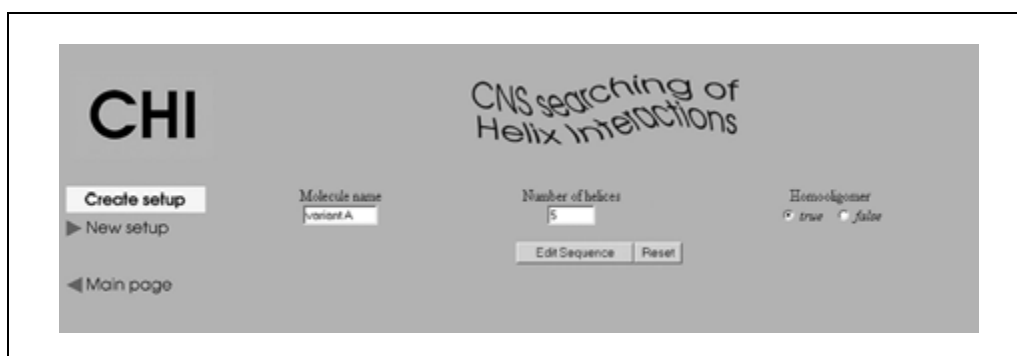


Figure 5.3.3 CHI "Create setup" first screen.

12a. Type the number of helices and choose the proper option between homo-oligomer false or true.

13a. Click "Edit sequence." A new editing screen will appear (Fig. 5.3.4).

14a. Type the first residue number, then enter the sequence in one-letter amino acid format (see APPENDIX 1A).

Note that the residue number is only important for the proper indexing of the sequence, and does not mean that the input sequence will be considered from that position.

15a. Choose the orientation of the helix.

If "true" was chosen for for homo-oligomer on the previous screen (step 12a), then one may choose either "up" or "down," as this option only describes the relative orientation between helices in a hetero-oligomer.

Clicking "View file" will allow one to view the chi_param that was created.

Choosing Edit file (see following) will allow one to edit all of the parameters in the chi_param file.

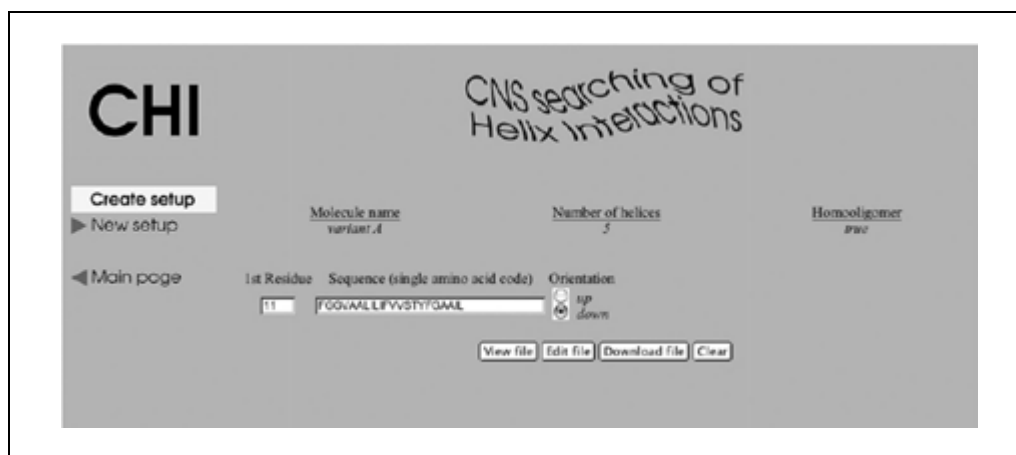


Figure 5.3.4 CHI “Create setup” Edit Sequence screen.

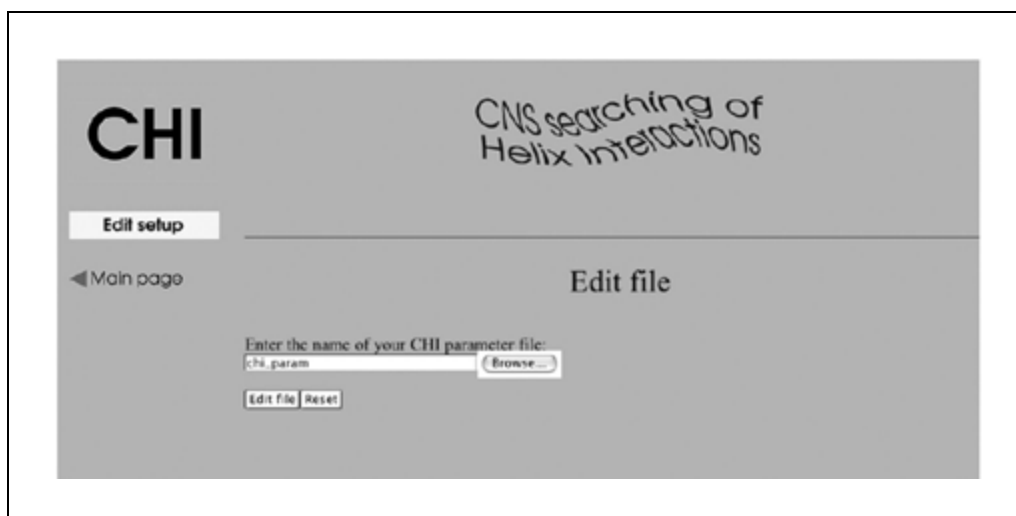


Figure 5.3.5 CHI “Edit setup” first screen for editing an existing parameters file.

To edit a parameter file that already exists

- 10b. In the CHI main menu on the left-hand side of the CHI home page (Fig. 5.3.2), click on “Edit setup.” In the first “Edit setup” screen that appears (Fig. 5.3.5), enter the full path and the name of the `chi_param` file, or click the Browse button, navigate to its location, and select it.
- 11b. Click on “Edit file.” Note the molecule structure parameters on the new screen that appears (Fig. 5.3.6):
 - Name of molecule
 - Number of helices
 - homo-oligomer, true/false.
- 12b. If one has chosen to simulate a hetero-oligomer, set the next parameters for each helix individually (otherwise they should only be set once):
 - Sequence
 - Residue number at start of sequence
 - Initial rotation offset around helix axis (the starting rotation angle about the helix axis relative to some arbitrary starting position, angle ϕ in Figure 5.3.1, default is 0.0°)

continued

Molecule structure	
name of the molecule	variantA
number of helices	5
homooligomer	☒ true ☐ false
molecular structure information for helix 1	
sequence	
PHE GLY GLY VAL ALA ALA LEU ILE LEU ILE PHE VAL VAL SER THR TYR PHE GLY ALA ALA ILE LEU	
More Lines	
residue number at start of sequence	11
initial rotation offset around helix axis	0.0
direction of helix	☐ up ☒ down
initial translational offset for helix along the z-axis	0.0
Search parameters	
extent of the search <i>a full search will sample all pairwise interactions a symmetric search will limit the search to symmetric pairs</i>	☐ full ☒ symmetric
search left handed crossing angles	☒ true ☐ false
search right handed crossing angles	☒ true ☐ false
type of molecular dynamics to use	☐ torsion ☒ cartesian
number of trials per structure	4
search parameters for helix 1	
rotation start (degrees)	0.0
rotation finish (degrees)	360.0
rotational step size (degrees)	10
Electrostatic effects	
value of dielectric constant	2
Initial rotation and tilt	
distance between centres of neighbouring helices - in Angstroms	10.4
left-hand crossing angle wrt diad axis - in degrees	25.0
right-hand crossing angle wrt diad axis - in degrees	-25.0
Clustering parameters	
cutoff for root mean square difference between two structures - Angstroms	1.25
minimum number of structures which define a cluster	9

Figure 5.3.6 CHI “Edit file” screen with structure parameters.

Direction of helix, up/down.

Initial translational offset for helix along the z axis (default is 0.0 Å).

13b. Set search parameters (Fig. 5.3.6):

i. Extent of the search: full or symmetric.

In a symmetric search, all of the helices rotate about their axis concomitantly ($\omega_i = \omega_j$) due to the symmetry assumption in homo-oligomers (Arkin, 2002). In a full search all rotation combinations are examined ($\omega_i \neq \omega_j$). The default for a homo-oligomeric complex is a “symmetric search” while a “full search” is the default when analyzing hetero-oligomers. A full search will obviously take much longer than a symmetric search due to the larger number of structures generated (see below).

ii. Search left-handed crossing angles, true/false (default is “True”).

iii. Search right handed crossing angles true/false (default is “True”).

iv. Type of molecular dynamics to use, torsion/cartesian (default is “torsion”).

The reader is referred to Rice and Brünger (1994) in order to evaluate which type of molecular dynamics to choose.

**Modeling
Structure from
Sequence**

5.3.7

- v. Number of trials per structure, i.e., number of searches to perform using different initial random velocities for each structure (default is 4).

If one has chosen to simulate a hetero-oligomer, the next parameters should be set for each helix individually; otherwise they should only be set once:

- vi. Rotation start (default is 0°).
- vii. Rotation finish (default is 360°).
- viii. Rotational step size (increment step default is 45°; the authors suggest setting it to 10° for a symmetric search).

- 14b. Set other restraints (it is not necessary to use these parameters in the Silent Amino Acid Substitution Protocol):

Electrostatic effects:

Value of the dielectric constant (for a membrane matrix enter 2.0; for a vacuum matrix enter 1.0)

Initial rotation and tilt:

Distance between centers of neighboring helices (default is 10.4 Å)

Left-hand crossing angle (default is 25°)

Right-hand crossing angle (default is -25°)

Clustering parameters:

Cutoff for root mean square difference between two structures (indicates structure similarity of two structures; default is 1 Å; a larger number would result in finding more clusters that are not as well grouped)

Minimum number of structures which define a cluster (default is 10).

- 15b. Click the “Save updated file” at the bottom of the screen, which will download a new, updated chi_param file into the local computer. Save it to the correct directory (e.g., ~/MyProtein/variantA/).

Run the GMDs search

- 16. Change to the correct working directory (e.g., ~/MyProtein/variantA) with the following command:

```
%> cd ~/MyProtein/variantA/
```

All commands should be issued from this directory unless noted otherwise.

- 17. In order to create the starting structure run:

```
%> chi_create -verbose
```

This is a fast process (taking a few minutes). The output files will be:

```
~/MyProtein/variantA/variantA.psf  
~/MyProtein/variantA/variantA.pdb  
~/MyProtein/variantA/chi_create.log  
~/MyProtein/variantA/results/create.out.
```

All of these files are accessory files that CHI uses. One might want to search for an error in the log file by issuing the following command:

```
%> grep -i err chi_create.log | more
```

18. Run the searching algorithm, to create all the structures, using the following command:

```
%> chi_search -verbose
```

The number of structures is:

$$\frac{\phi_{\text{end}} - \phi_{\text{start}}}{\text{increment}} \times \text{handedness} \times \text{trials} =$$
$$\frac{360^\circ - 0^\circ}{10} \times 2 \times 4 = 288$$

This process is time-consuming (typically many hours). As an example, simulating a bundle of 5 helices, each composed of 28 amino acids, takes ~20 to 30 min per structure, on a DEC Alpha 433 AU (a relatively slow machine nowadays).

This will produce the following output:

```
~/MyProtein/variantA/results/search.out
```

which contains the results of the simulation (energy and orientational parameters for each structure) and the pdb for each structure simulated.

The names of the pdb files are as follows:

```
~/MyProtein/variantA/results/left_i_j.pdb
```

where i is the initial angle of rotation and j is the trial number.

Right-handed structures will be designated similarly:

```
~/MyProtein/variantA/results/right_i_j.pdb
```

One may wish to check for errors during the run by screening the log file with the following command:

```
%> grep -i err chi_search.log | more
```

If no errors are found, it is best to delete the log files since they can be very large (several megabytes).

19. To calculate the C_α RMSD between all of the structures, type:

```
%> chi_rmsd -verbose
```

This process is relatively time-consuming (roughly 0.1 sec per comparison, i.e., 93 min for 288 structures). The output is a single file:

```
~/MyProtein/variantA/results/rmsd.out.
```

This file contains a list of structures and the C_α RMSDs between them. Note that the file only lists those structure that are lower than the RMSD threshold plus 1 Å.

Note that when the number of structures increases to a certain point, it is the RMSD calculation that consumes the largest amount of CPU time. This is because the time required for molecular dynamics simulations scales linearly with the number of structures generated (576 structures take only twice the amount of time as 288 structures), whereas the RMSD calculation scales with the square of the number of structures (comparison of 576 structures takes 4 times longer than 288 structures). The chi_rmsd script is therefore best suited to cases where the number of structures is approximately 2000 or less. If interested in simulating a larger system, one can contact the authors (arkin@cc.huji.ac.il) for alternative scripts to chi_rmsd. These scripts reduce the computational cost by not calculating the RMSD between two clusters if their orientational parameters differ markedly.

20. In order to search for clusters to which structures have converged, run the following command:

```
%> perl /usr/local/chi/bin/ak_cluster.pl
```

This file is different from the clustering file in the CHI package (chi_cluster) in that all structures that it places in a cluster are similar to one another. In chi_cluster, all structures are similar to at least one structure, but not necessarily to all of them. This step is very fast (taking a few seconds).

21. Using any text editor, view the output file, ~/MyProtein/variantA/results/cluster.out, to see how many clusters were obtained.

The authors recommend creating at least 10 to 15 clusters for each variant in order to find a “complete set” for all the variants (see below). This can be achieved by empirically changing the clustering parameters in the file ~/MyProtein/variantA/chi_param, either using a text editor or through the CHI Web interface (see above). There are two methods for increasing the number of clusters: (1) relaxing the RMSD threshold (i.e., increasing it) and (2) decreasing the required number of structure per cluster. Both methods should be tried.

22. In order to calculate an average, representative structure for each cluster, issue the following command:

```
%> chi_average
```

This process is moderately time consuming (taking a few hours). The output is both a file depicting the results of the program:

```
~/MyProtein/variantA/results/average.out
```

which includes the orientational parameters and energy of each cluster average, and the structure for each cluster average:

```
~/MyProtein/variantA/results/clusterN.pdb
```

where N is the number of the cluster.

Find a “complete set”

23. Repeat GMDs (steps 16 to 22) for all the variants.

Remember that each variant’s search should be undertaken in its specific subdirectory, e.g.:

```
~/MyProtein/variantB/, ~/MyProtein/variantC/)
```

The following steps are in preparation for comparing clusters of different variants and are not part of the “standard” CHI package. The process starts with creating a virtual GLY variant. Selecting the right cluster, which is the one that exists in all the variants, depends on comparing the RMSD between all the structures obtained in the previous steps. Comparing RMSD for all the atoms of every two variants is impossible, due to the fact that they differ at one or more of their amino acids. However, one may avoid this problem by comparing only the RMSD of their backbones. Therefore, a virtual variant, whose sequence is composed only of glycine should be created.

24. Create a new subdirectory named GLY in the upper directory, using, e.g., the command:

```
~/MyProtein/GLY/
```

25. Create a new chi_param file (also see steps 10a to 15a and 10b to 15b) in which the molecule name is GLY and all the amino acids are glycine (Fig. 5.3.7). Leave all other parameters exactly as they are in all the other variants’ parameter files (including the length of the sequence). Save that file in ~/MyProtein/GLY/ directory.

Figure 5.3.7 Creating a glycine parameter file.

26. Change directory:

```
%> cd ~/MyProtein/GLY/
```

and run:

```
%> chi_create
```

This will create the files ~/MyProtein/GLY/GLY.pdb and ~/MyProtein/GLY/GLY.psf. See step 24 annotation for explanation.

27. Change directory to the parent directory:

```
%> cd ~/MyProtein/
```

and copy the following files:

```
%> cp ~/MyProtein/GLY/GLY.p* ~/MyProtein/
```

28. Create a variants list, i.e., edit a text file named `list` (not `list.txt`) that will contain the names of all the variants subdirectories.

Each line should contain only a single variant. An example of the content of such a file with three variants is listed below:

```
variantA
variantB
variantC
```

Save the `list` file to the upper directory (i.e., `~/MyProtein/list`).

29. Copy and paste the following file:

```
%> cp /usr/local/chi/bin/cns.inp ~/MyProtein/
```

30. Check that the parent directory, `~/MyProtein/` contains the appropriate files by issuing the following command:

```
%> ls ~/MyProtein/
```

A typical directory listing with three homologs should be:

continued

**Modeling
Structure from
Sequence**

5.3.11

```
variantA/  
variantB/  
variantC/  
GLY/  
GLY.pdb  
GLY.psf  
cns.inp  
list
```

31. Compare all the cluster averages from each homolog, obtained by GMDS by their C_{α} RMSD. Look for the cluster that exists in all variants with a minimal RMSD between every pair of variants. Issue all of the following commands from the parent directory, ~/MyProtein/. To be sure that one is in the right directory, issue the following command:

```
%> cd ~/MyProtein/
```

Run the following command to compare the different homologs:

```
%> perl /usr/local/chi/bin/compare_rmsd.pl N
```

where N is a number that signifies the RMSD threshold in Å.

There are several output files:

~/MyProtein/compare_rmsd.out: This file contains the list of clusters that are found in all of the homologs. If more than one structure is found, try to enforce a stricter threshold by reducing the number.

~/MyProtein/cns_rmsd.result: This file lists all pairwise RMSD results.

~/MyProtein/rmsd_calculation_list: This is an accessory file to be used by CNSsolve.

~/MyProtein/log: This is the CNSsolve log file.

As stated above, view the file compare_rmsd.out in order to decide whether to repeat the previous step with a different threshold or not.

32. Repeat the above steps until a single cluster is identified that is found in all variants.

GUIDELINES FOR UNDERSTANDING RESULTS

The procedure outlined in the Basic Protocol is a relatively simple one that involves two steps: (1) generate possible structures for each of the variants and (2) check if there is one structure that persists in all of the different variants. There are few key points to which one should pay close attention, and these are outlined below.

How Well Do the Individual Variants Cluster?

The clustering parameters (i.e., the RMSD threshold and the minimal number of structure per cluster) are chosen arbitrarily. They will obviously change the outcome of the simulation in that they will change the number of possible structures from each variant. The authors have tended not to extend the RMSD threshold beyond 1.25 Å, and the number of structures per cluster is not lowered beyond 7.

How Well Does a Single Structure Persist in All of the Variants?

As stated in the above section, it is difficult to place an upper boundary on the RMSD threshold, stating unequivocally that above a certain limit the structures are no longer the same. One should keep in mind however that RMSD is obviously not a linear repre-

sensation of similarity. In other words, when the RMSD between two structures is 1 Å instead of 2 Å, they have not become twice as similar. Empirically, the authors tend not to raise the RMSD threshold beyond 1.5 Å.

COMMENTARY

Background Information

Structural studies have so far shown that membrane proteins fold into one of only two topologies: β -barrels or α -helical bundles. Since α -helical membrane proteins are far more abundant, as well as pharmaceutically more important, the following discussion will be restricted to this family.

Predicting membrane protein structure is of significant importance because, despite the pharmaceutical importance that they possess, out of nearly 20,000 protein structures solved using crystallographic or NMR methods, only a few dozen are membrane proteins. This paucity of experimentally solved structures is striking, considering that according to a recent census of genomes, 20% to 30% of all genes are predicted to encode membrane proteins (Stevens and Arkin, 2000).

Knowledge-based homology methods that rely on structural information are difficult to implement for membrane proteins, simply because of the lack of solved structures. On the other hand, other modeling methods are relatively easy to implement for membrane proteins, compared to water-soluble proteins, due to the overall simplicity of membrane proteins, in particular those formed from α -helical bundles. Furthermore, assignment of the different helices in an α -helical bundle (the more abundant and pharmaceutically important family) is relatively straightforward. Thus, it can be concluded that while the structures of α -helical membrane proteins are the most difficult to determine experimentally, fortunately they are the easiest to predict computationally.

Despite the apparent ease with which it is possible to simulate membrane proteins using molecular dynamics, there is one issue that has can potentially present difficulty: the presence or absence of a lipid bilayer. In the simulations of membrane proteins using molecular dynamics in CHI, no lipids or solvent molecules are employed, because of the prohibitive computational cost. However it is possible to argue that the most important stabilizing force in any oligomeric bundle will be the interaction between the helices themselves (Torres et al., 2001). Thus, there is some justification in the simulation procedure described here, although

the lack of a lipid environment should always be borne in mind.

Critical Parameters and Troubleshooting

The underlining premise of GMDS is that it is possible to exhaustively search the configuration space of a transmembrane helical bundle and come up with several candidate structures. One of these structures is presumed to be that which is found in nature.

The underlining premise of silent substitution modeling is that silent substitutions do not disrupt the native structure, but may destabilize non-native structures. Thus, it is possible to select the correct structure among several candidate structures using silent substitution modeling by looking for a model that is present in all of the homologs.

When will this procedure fail? There are several possible situations in which this may occur: (1) where no single structure is found to be in all of the homologs; (2) where more than one structure is found in all of the homologs; and (3) where the structure that is found in all of the sequences is not the native one. Below, the potential causes of these failures are analyzed and the ways to avoid them are suggested.

No structure is found

There can be two simple reasons for the failure to find a structure that persists in all of the homologs.

GMDS was not able to identify the native structure in at least one homolog and perhaps in all of them. The authors have found from experience that this may happen when the tilt of the helices is relatively large, as is the case in the Influenza A M2 H⁺ channel (Kukol et al., 1999). This problem may be overcome by increasing the crossing angle used for the right and left-handed searches from 25° by editing the `chi_param` file. Other options that may be pursued are to increase the number of trials and to reduce the rotational increment (again in `chi_param`). Obviously both of these changes will be reflected in increased computational time.

Some of the mutations are not silent. In other words, some of the homologs do not adopt the

same structure (Torres et al., 2002a). Here the authors suggest an increase in the similarity threshold of the sequences used in the simulation, i.e., sequences that are closer to the target protein have a better chance of adopting the same structure.

More than one structure is found

In this instance, it is possible that the “filtering capabilities” of the silent mutant were not sufficient. The recommendation is simply to use more sequences, by potentially lowering the identity threshold.

The structure found is incorrect

In all of the cases in which the authors have used the combination of GMDS and silent amino acids substitution modeling, it produced the correct structure as verified using other experimental methods (Kukul et al., 2002; Torres et al., 2002b; Torres et al., 2002c). However, this may not always be the case. Identifying such a situation is difficult and requires the application of potentially time-consuming experimental methodologies (see Suggestions for Further Analysis).

Suggestions for Further Analysis

It is obvious that the best way to analyze the results of any modeling exercise is by experimentation. There are several methods that can be applied; however most experiments, short of directly solving the structure, are better suited to refuting models, rather than confirming them. The reason is that, typically, more than one model can be consistent with the experimental results.

Mutagenesis

Mutagenesis has been used in several instances to determine which residues are essential for oligomerization of particular transmembrane helices. This is possible only when an oligomerization assay exists, as with glycophorin A, which remains dimeric in SDS-PAGE (Lemmon et al., 1992a). In that series of experiments, several residues were identified that were shown to line one side of a helix projection (Lemmon et al., 1992b; Lemmon et al., 1994). A solution NMR study in detergent micelles has shown those residues to be intimately involved in the helix-helix interface (MacKenzie et al., 1997). Mutagenesis has also been performed for phospholamban, which also remains a pentamer in SDS-PAGE (Arkin et al., 1994). In this instance, however, more than one model was consistent with the mutagenesis

results and only a direct structural method was able to resolve this ambiguity (Torres et al., 2000).

Literature Cited

- Adams, P.D., Arkin, I.T., Engelman, D.M., and Brünger, A.T. 1995. Computational searching and mutagenesis suggest a structure for the pentameric transmembrane domain of phospholamban. *Nat. Struct. Biol.* 2:154-162.
- Arkin, I.T. 2002. Structural aspects of oligomerization taking place between the transmembrane alpha-helices of bitopic membrane proteins. *Biochim. Biophys. Acta* 1565:347-363.
- Arkin, I.T., Adams, P.D., MacKenzie, K.R., Lemmon, M.A., Brünger, A.T., and Engelman, D.M. 1994. Structural organization of the pentameric transmembrane alpha-helices of phospholamban, a cardiac ion channel. *EMBO J.* 13:4757-4764.
- Brünger, A.T., Adams, P.D., Clore, G.M., DeLano, W.L., Gros, P., Grosse-Kunstleve, R.W., Jiang, J.S., Kuszewski, J., Nilges, M., Pannu, N.S., Read, R.J., Rice, L.M., Simonson, T., and Warren, G.L. 1998. Crystallography & NMR system: A new software suite for macromolecular structure determination. *Acta Crystallogr. D Biol. Crystallogr.* 54:905-921.
- Kukul, A., Adams, P.D., Rice, L.M., Brünger, A.T., and Arkin, T.I. 1999. Experimentally based orientational refinement of membrane protein models: A structure for the Influenza A M2 H⁺ channel. *J. Mol. Biol.* 286:951-962.
- Kukul, A., Torres, J., and Arkin, I.T. 2002. A structure for the trimeric MHC class II-associated invariant chain transmembrane domain. *J. Mol. Biol.* 320:1109-1117.
- Lemmon, M.A., Flanagan, J.M., Hunt, J.F., Adair, B.D., Bormann, B.J., Dempsey, C.E., and Engelman, D.M. 1992a. Glycophorin A dimerization is driven by specific interactions between transmembrane alpha-helices. *J. Biol. Chem.* 267:7683-7689.
- Lemmon, M.A., Flanagan, J.M., Treutlein, H.R., Zhang, J., and Engelman, D.M. 1992b. Sequence specificity in the dimerization of transmembrane alpha-helices. *Biochemistry* 31:12719-12725.
- Lemmon, M.A., Treutlein, H.R., Adams, P.D., Brünger, A.T., and Engelman, D.M. 1994. A dimerization motif for transmembrane alpha-helices. *Nat. Struct. Biol.* 1:157-163.
- MacKenzie, K.R., Prestegard, J.H., and Engelman, D.M. 1997. A transmembrane helix dimer: Structure and implications. *Science* 276:131-133.
- Rice, L.M. and Brünger, A.T. 1994. Torsion angle dynamics: Reduced variable conformational sampling enhances crystallographic structure refinement. *Proteins* 19:277-290.
- Stevens, T.J. and Arkin, I.T. 2000. Do more complex organisms have a greater proportion of membrane proteins in their genomes? *Proteins* 39:417-420.

- Torres, J., Adams, P.D., and Arkin, I.T. 2000. Use of a new label, $^{13}\text{C}=^{18}\text{O}$, in the determination of a structural model of phospholamban in a lipid bilayer: Spatial restraints resolve the ambiguity arising from interpretations of mutagenesis data. *J. Mol. Biol.* 300:677-685.
- Torres, J., Kukol, A., and Arkin, I.T. 2001. Mapping the energy surface of transmembrane helix-helix interactions. *Biophys. J.* 81:2681-2692.
- Torres, J., Briggs, J.A., and Arkin, I.T. 2002a. Contribution of energy values to the analysis of global searching molecular dynamics simulations of transmembrane helical bundles. *Biophys. J.* 82:3063-3071.
- Torres, J., Briggs, J.A., and Arkin, I.T. 2002b. Convergence of experimental, computational and evolutionary approaches predicts the presence of a tetrameric form for CD3-zeta. *J. Mol. Biol.* 316:375-384.
- Torres, J., Briggs, J.A., and Arkin, I.T. 2002c. Multiple site-specific infrared dichroism of CD3-zeta, a transmembrane helix bundle. *J. Mol. Biol.* 316:365-374.
- Treutlein, H.R., Lemmon, M.A., Engelman, D.M., and Brünger, A.T. 1992. The glycoporphin A transmembrane domain dimer: Sequence-specific propensity for a right-handed supercoil of helices. *Biochemistry* 31:12726-12732.

Key References

Arkin et al., 1994. See above.

In this article, global searching molecular dynamics simulation is used to find a model for phospholamban.

Adams et al., 1995. See above.

Here, the theory of global searching molecular dynamics simulation is presented in detail.

Briggs, J.A.G., Torres, J., Kukol, A., and Arkin, I.T. 2001. A new method to model membrane protein structure based on silent amino-acid substitutions. *Proteins Struct. Funct. Genet.* 44:370-375.

In this article, silent substitution modeling is introduced for the first time.

Torres et al., 2002a. See above.

In this paper, results of global searching molecular dynamics simulations are analyzed in terms of energy, thereby enabling the user to further select among candidate models.

Torres et al., 2002b. See above.

In this work, silent substitution modeling is employed to derive a structure of the TCR CD3 ζ transmembrane helical bundle, shown to coincide with that obtained experimentally.

Contributed by Uzi Kochva, Hadas Leonov,
and Isaiah T. Arkin
The Hebrew University
Jerusalem, Israel

Paul D. Adams
Lawrence Berkeley Laboratory
Berkeley, California

Representing Structural Information with RasMol

UNIT 5.4

Thousands of atomic structures of proteins, nucleic acids, and other biomolecules are available for use in research and education. Many effective tools are available for the display of these structures. These tools run on popular computer hardware, and they provide a standard set of options for representation of the molecule. This unit will describe the use of a common program, RasMol, for the display of molecular structures. RasMol is simple to get started and provides a wide range of options as one explores a molecular structure. Many of the principles of selection and display used in RasMol will then be directly applicable when moving to other molecular graphics programs for specific applications.

The unit will begin with the basics of obtaining coordinates and displaying them in RasMol (Basic Protocol 1). Next, the advantages and limitations of different representations will be discussed (Alternate Protocol). A common pitfall encountered in the display of atomic coordinates, obtaining the proper biological unit, will be presented (Basic Protocol 2). Finally, some ideas for customizing a molecular graphics session will be presented (Basic Protocol 3).

USING RasMol TO DISPLAY A PROTEIN STRUCTURE

BASIC PROTOCOL 1

In this protocol, the coordinates of hemoglobin will be downloaded from the Protein Data Bank and the structure displayed in RasMol using a few basic representations. RasMol is an open-source program designed for the display of biological molecules. The program reads molecular coordinates from a file and interactively displays the molecule in a variety of representations. RasMol is an excellent place to start when learning about molecular graphics, since the program has a number of useful options available in convenient pull-down menus. Then, as further functionality is needed for specific applications, the Command Line interface allows additional selection and representation options.

Necessary Resources

Hardware

RasMol runs on a variety of computer hardware, including personal computers.

Software

Operating system: RasMol runs under Microsoft Windows and Apple Macintosh OS 7.0 or higher (including Mac OS X). It may also be run on workstations under Unix, Linux, or VMS.

RasMol: Binary versions of RasMol are available on the WWW at: <http://www.bernstein-plus-sons.com/software/rasmol/>. Downloading and installation instructions are given in Support Protocol 1.

Files

Coordinate files are read in a variety of formats, including PDB, Mol2, CHARMm, and mmCIF. The program deals gracefully with a number of variations of these files, including files containing coordinates for multiple conformers or multiple models. In this example, coordinates for hemoglobin (2HHB.pdb) obtained from the Protein Data Bank (PDB; UNIT 1.9) are used; instructions for downloading the PDB coordinate file are given in Support Protocol 2.

Modeling Structure from Sequence

Contributed by David S. Goodsell

Current Protocols in Bioinformatics (2005) 5.4.1-5.4.23

Copyright © 2005 by John Wiley & Sons, Inc.

5.4.1

Supplement 11

Display hemoglobin with RasMol

1. Download and install RasMol on the local machine (Support Protocol 1).
2. Download the PDB coordinate file 2HHB .pdb from the PDB (UNIT 1.9) as described in Support Protocol 2.
- 3a. *On Unix and Linux machines:* Type `rasmol 2HHB.pdb` at the prompt. This will start RasMol and load the coordinates from the file 2HHB.pdb.
- 3b. *On personal computers:* Double-click on the RasMol icon. This will launch RasMol. Next, select Open from the File menu to load the coordinates from the file 2HHB.pdb. Finally, go to the Window menu and select the Command Line window. This will open a new window that contains the Command Line interface. With the Mac OS X operating system one can also simply drag the icon for the desired PDB file on to the icon for RasMol and the program will automatically open, load the coordinates, and display the default wireframe representation.

At this point, the screen should look like Figure 5.4.1, with a graphics window that shows the hemoglobin structure and a Command Line window. The molecule may be rotated by holding down the mouse button in the graphics window and dragging the cursor in different directions. Other transformations, such as scaling the image to different sizes and translating the molecule to different locations in the screen, are accessible through different buttons (if using a three-button mouse) and through combinations of holding the mouse button and depressing the Shift or Control keys (if using a one-button mouse).

4. The pull-down menus in the graphics window make it possible to change the representations used to display the molecule, as well as to change some common parameters used to create the display.
 - a. In the Display menu, several common representations of the structure may be chosen, as described more fully in the Alternate Protocol.

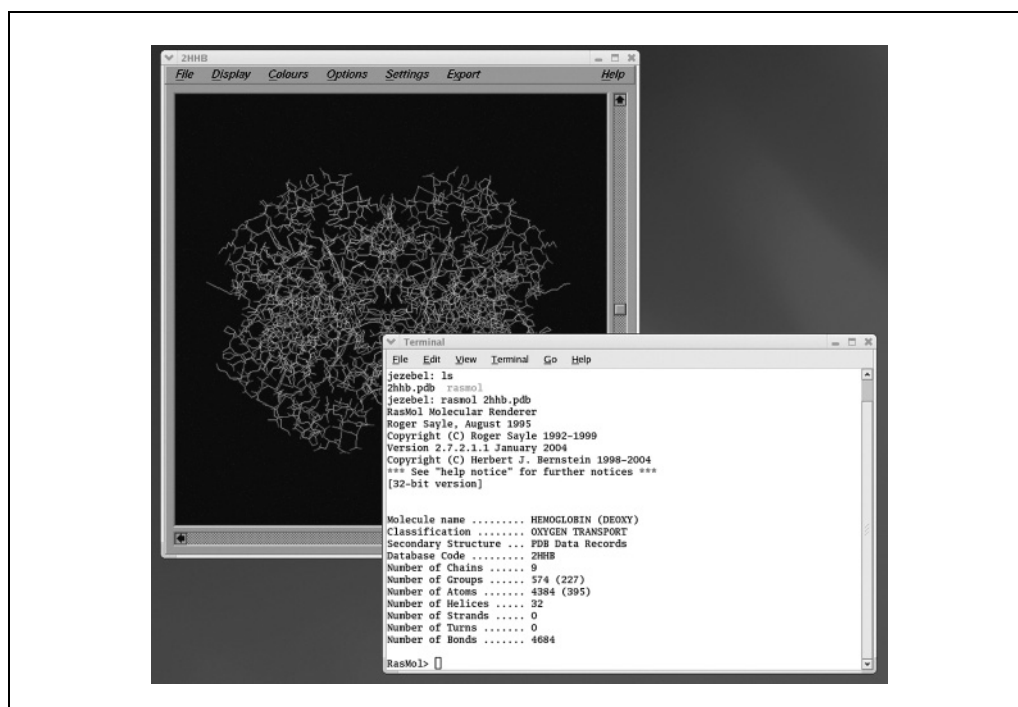


Figure 5.4.1 RasMol running on the computer display. The viewer window is at upper left, behind the Command Line window at lower right.

- b. Using options in the Colours menu, the structure may be colored using traditional atomic colors or several other schemes that highlight different characteristics.
 - c. In the Options menu, slab mode may be used to cut away the nearest portions of the molecule, and specular highlights and shadows may be toggled on and off.
 - d. The Settings menu makes it possible to choose an action that will be performed when clicking on a portion of the molecule, e.g., measuring distances between atoms.
 - e. Finally, the Export menu makes it possible to save images from the graphics window.
5. The Command Line window (labeled “Terminal” in Fig. 5.4.1) allows direct control of all of the commands available in RasMol. A few of the most common commands will be used in this unit. The user manual contains instructions for using a variety of other specialized commands once the basics are mastered.

Take some time to explore the options available in the pull-down menus and to become familiar with manipulating the molecule. When ready to move on to the next step, quit the program by typing, in the Command Line window:

```
RasMol> quit
```

The remainder of this protocol will discuss a few useful representations and provide a few tips to solve common problems. The Command Line window will be used to change representations and colors, allowing more control than that available through the pull-down menus.

Representations and their uses

Three basic types of representations are commonly used to display biological molecules. Each has its own strengths and weaknesses, and each is designed for a specific use.

6. *Wireframe diagrams:* The default representation in RasMol is a wireframe diagram. Each line represents a covalent bond between atoms. This representation is ideal for examination of the atomic details of the structure. However, wireframe representations tend to be very complicated. This is acceptable when examining the structure interactively, but wireframe representations are generally too crowded for printed images. The following describes how to create a wireframe diagram.

- a. Restart RasMol using the 2hhb coordinates.
- b. In the Command Line window, type the following series of commands:

```
RasMol> select HEM
```

This selects the heme group.

```
RasMol> wireframe 150
```

This represents the heme with a thick wireframe; values go from 1 (thin) to 500 (thick).

```
RasMol> select iron
```

This selects the iron ion.

```
RasMol> cpk 150
```

This represents the iron as a sphere. The command cpk, which represents atoms as spheres, refers to the plastic Corey-Pauling-Koltun models used for building small organic molecules, which were the first models that used a spacefilling representation. The units used by RasMol are integers that correspond to 1/250th of an Angstrom (Å).

The display should look like Figure 5.4.2. The protein is displayed with a wireframe, colored by the atom type, and thicker bonds are used to make the heme group more apparent.

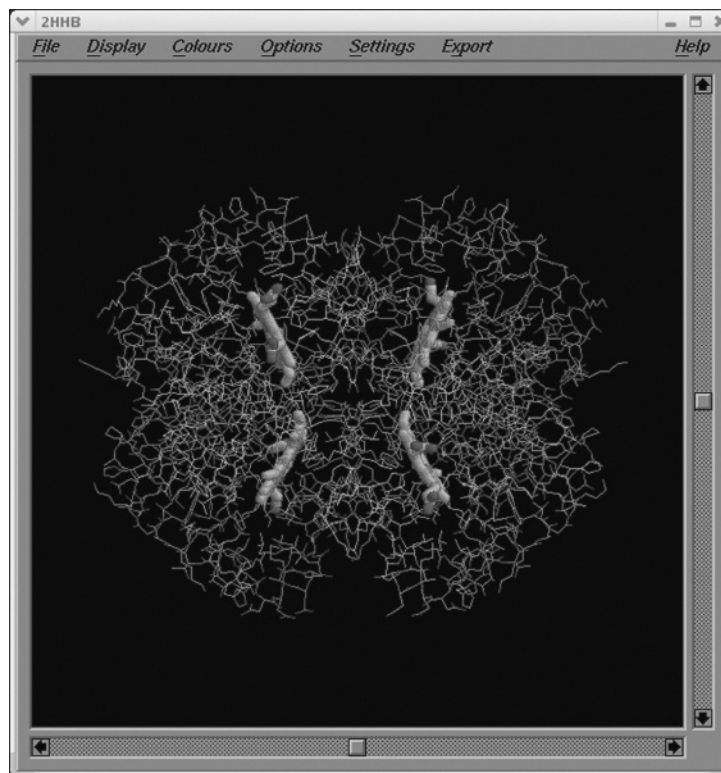


Figure 5.4.2 Hemoglobin with the heme groups in thick bonds and the iron ions shown as small spheres.

- c. Rotate the display and notice the following. (1) Individual amino acids may be identified from their shape and chemical composition. For instance, look for aromatic amino acids while rotating the structure. (2) The overall conformation of the backbone is difficult to comprehend. Wireframe images often look like a tangle of atoms, not a folded chain. (3) Zoom the molecule to higher magnification and notice that the wireframe works best on close-up pictures, which focus on a few details.
7. *Spacefilling diagrams*: Spacefilling representations show the size and shape of the entire molecule. Each atom is represented by a sphere that represents the optimal contact distance between nonbonded atoms. The following describes how to create a spacefilling diagram.

- a. In the Command Line window, type the following series of commands:

```
RasMol> select all
```

This selects all atoms.

```
RasMol> wireframe off
```

This turns off the wireframe.

```
RasMol> select protein or ligand
```

This selects only the protein atoms and the ligand (heme) atoms.

```
RasMol> cpk
```

This displays atoms as spheres, using the default radius for the spheres.

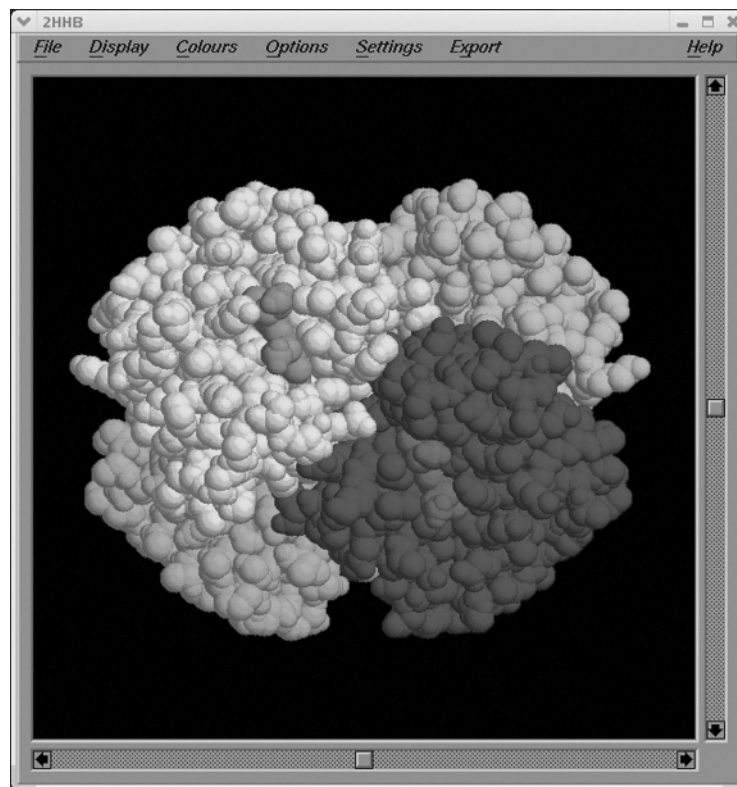


Figure 5.4.3 Spacefilling (cpk) representation of hemoglobin with each chain colored differently. For the color version of this figure go to <http://www.currentprotocols.com>.

```
RasMol> color chain
```

This colors each chain a different color.

The display should look like Figure 5.4.3. Now, the entire protein is displayed as space-filling spheres for each atom. The four individual polypeptide chains that make up the hemoglobin tetramer are each given a different color. The logical operation used in the selection command is a typical Boolean OR, so the command "select protein or ligand" will select all atoms in the protein and all atoms in the ligand. Similarly, the command "select protein and ligand" will select no atoms, since there are no common atoms that are in both the set of protein atoms and the set of ligand atoms. Selection of an appropriate set of atoms is probably the most difficult, and the most useful, aspect of RasMol usage.

- b. Rotate the display and notice the following. (1) Spacefilling representations show the bulk of the protein. Notice the way the different subunits interdigitate, and the way the heme slots into a form-fitting groove. (2) Many people find it difficult to identify individual amino acids in spacefilling representations, even if they are colored by atom type.
8. **Backbone and Ribbon Diagrams:** Two schematic representations are commonly used to display the topology of a protein chain. In a backbone representation, cylinders are drawn between successive alpha carbon positions. In a ribbon diagram, a helical ribbon is used to display alpha helices, a large flat arrow is used to display beta sheets, and smooth tubes are used to display other portions of the chain. Ribbon diagrams are excellent for presentation of protein folding, and are currently the most common representation used in journal publications. The following describes how to create backbone and ribbon diagrams.

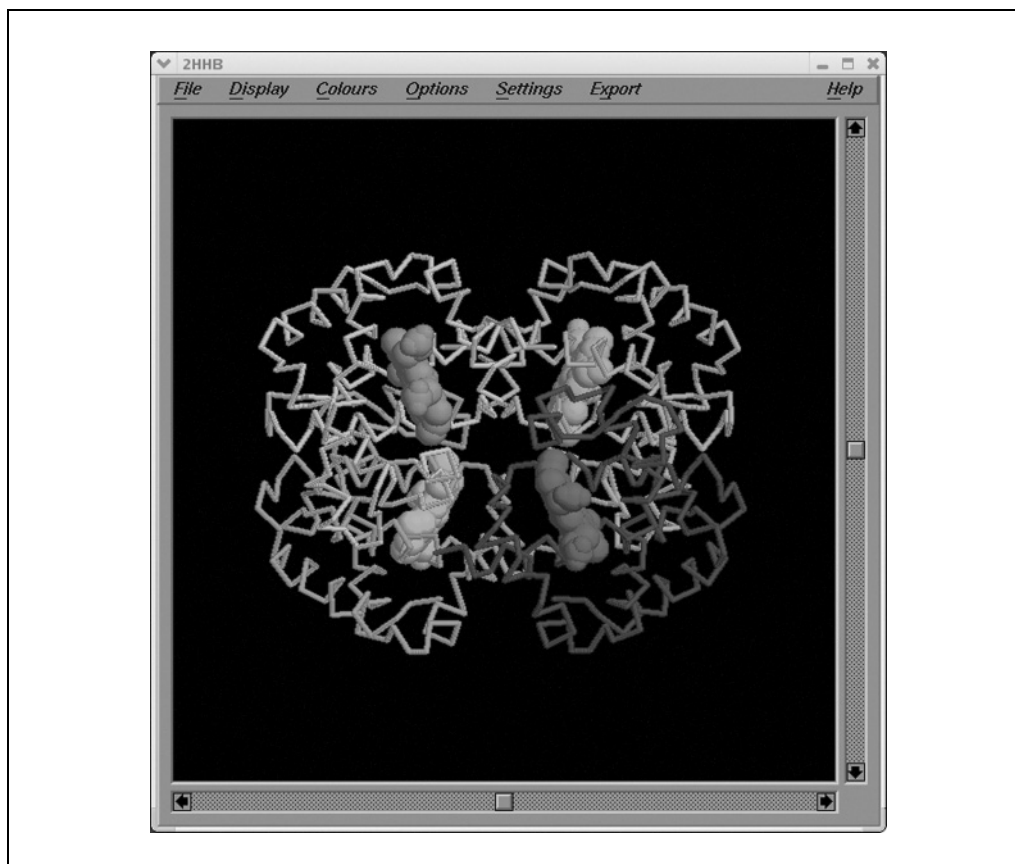


Figure 5.4.4 Backbone representation of the hemoglobin protein chains, with the hemes still shown as spacefilling spheres. For the color version of this figure go to <http://www.currentprotocols.com>.

- a. In the Command Line window, type:

```
RasMol> select protein
```

This selects the protein.

```
RasMol> cpk off
```

This turns off the spheres (the spheres for the heme remain on).

```
RasMol> backbone 100
```

This draws a tube along the backbone.

The display should look like Figure 5.4.4.

- b. Rotate the display and notice the following. (1) Backbone representations show the folding of the protein chain, making it easy to recognize the many alpha helices in this globin fold. (2) Backbone representations typically under-represent the size of the protein, and ignore the dense packing of atoms in the structure. Explore this by flipping the spacefilling representation on and off by typing `cpk` and then `cpk off` in the Command Line window. (3) The position of each alpha carbon is retained in the diagram, so it is possible to identify the location of each amino acid.
- c. Next, in the Command Line window, type the following commands:

```
RasMol> backbone off
```

This turns off the protein backbone.

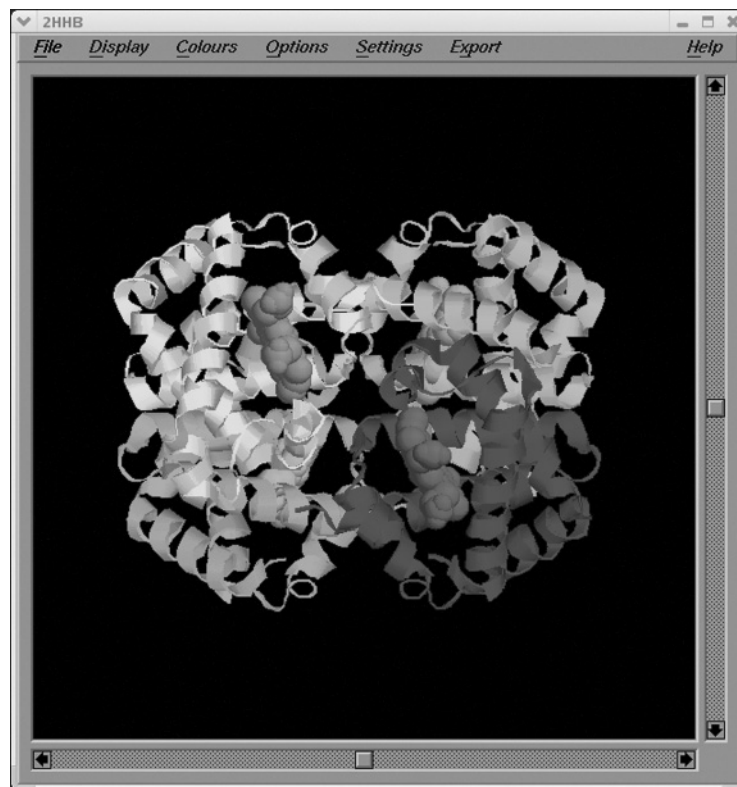


Figure 5.4.5 Ribbon diagram (cartoon) of the hemoglobin protein chains, with the hemes as spacefilling spheres. For the color version of this figure go to <http://www.currentprotocols.com>.

```
RasMol> cartoon
```

This turns on the ribbon diagram.

The display should look like Figure 5.4.5.

- d. Rotate the display and notice the following. (1) Ribbon diagrams make it easy to identify secondary structural elements, such as the alpha helices in hemoglobin. (2) Visual cues to amino acid positions are lost in the smooth ribbon, unless the ribbon is colored to show the types of amino acids.
9. When finished, type the following in the Command Line window to exit the program:

```
RasMol> quit
```

DOWNLOADING AND INSTALLING RasMol ON A LOCAL COMPUTER

This protocol describes how to download and install RasMol on a local computer. Executable versions of RasMol are available on the WWW, so this is relatively straightforward.

Necessary Resources

Hardware

RasMol runs on a variety of computer hardware, including personal computers.

SUPPORT PROTOCOL 1

Modeling Structure from Sequence

5.4.7

Software

Operating system: RasMol runs under Microsoft Windows and Apple Macintosh OS 7.0 or higher (including Mac OS X). It may also be run on workstations under Unix, Linux, or VMS.

Browser: An Internet browser is required

1. Point the browser to <http://www.bernstein-plus-sons.com/software/rasmol/>.
2. Click on the appropriate version at the top of the page to download the executable file.

On personal computers, the program will appear as a RasMol icon. On Linux machines, the program will appear as a file with a name like rasmol.8BIT or rasmol.32BIT.

3. On workstations, ensure that the permission is set correctly for an executable file, for instance, with the command:

```
chmod a+x rasmol.32BIT
```

DOWNLOADING COORDINATES FROM THE PROTEIN DATA BANK

The Protein Data Bank (*UNIT 1.9*) is the primary repository of protein structure data. It is designed for easy searching and downloading. This protocol describes how to download the coordinates of hemoglobin.

Necessary Resources

Hardware

The Protein Data Bank on a variety of computer hardware, including personal computers

Software

An Internet browser is required.

1. On the main PDB WWW page (<http://www.pdb.org>), type 2hhb in the Search the Archive box, then hit the Search button.

This will load the Structure Explorer page for the structure.

2. Click on the Download/Display File link on the left side.
3. Click on the link for “complete with coordinates” in the “PDB” and “TEXT” format.
4. Click the “Save full entry to disk” button. This will download the file 2HHB.pdb to the local computer.

Coordinates for thousands of other biomolecules at the Protein Data Bank may be accessed in a similar way. On the main PDB WWW page, one may use the Search the Archive box to search the database using the names of molecules, authors, molecule types, and a variety of different searching options.

TWO USEFUL VIEWS IN RasMol

This protocol includes two quick methods for creating RasMol images that fill specific needs. The first method provides a fast overview of the structure, making it possible to see the major structural features when exploring a new protein. The second method makes it possible to pinpoint key amino acids within a complex protein structure.

Necessary Resources

Hardware

RasMol runs on a variety of computer hardware, including personal computers.

Software

Operating system: RasMol runs under Microsoft Windows and Apple Macintosh OS 7.0 or higher (including Mac OS X). It may also be run on workstations under Unix, Linux, or VMS.

RasMol: Binary versions of RasMol are available on the WWW at: <http://www.bernstein-plus-sons.com/software/rasmol/>. Downloading and installation instructions are given in Support Protocol 1.

Files

Coordinate files are read in a variety of formats, including PDB, Mol2, CHARMm, and mmCIF. The program deals gracefully with a number of variations of these files, including files containing coordinates for multiple conformers or multiple models. In this example, coordinates for hemoglobin (2HHB.pdb) obtained from the Protein Data Bank (PDB; UNIT 1.9) are used; instructions for downloading the PDB coordinate file are given in Support Protocol 2.

An overview representation

This representation is useful for the first look at a protein, to provide a quick understanding of the overall shape, the number of chains and how they are folded, and the location of any ligands or prosthetic groups. This representation is also commonly used in publications to give an overall summary of the structure of the protein. This overview representation will display the protein chains as backbones (or ribbons, if preferred), with different colors on each chain. The ligands are drawn with spacefilling spheres to make them easy to find.

1. Restart RasMol with the 2hhb coordinate set (see Basic Protocol 1).

This will give the wireframe representation.

2. In the Command Line window, type the following series of commands:

```
RasMol> wireframe off
```

This turns off the default representation.

```
RasMol> select ligand
```

This selects just the ligand.

```
RasMol> cpk
```

This displays the ligand with spheres.

```
RasMol> select protein
```

This selects just the protein.

```
RasMol> backbone 100
```

This displays the protein with a thick backbone.

```
RasMol> color chain
```

This colors each chain a different color.

The display should look like Figure 5.4.6.

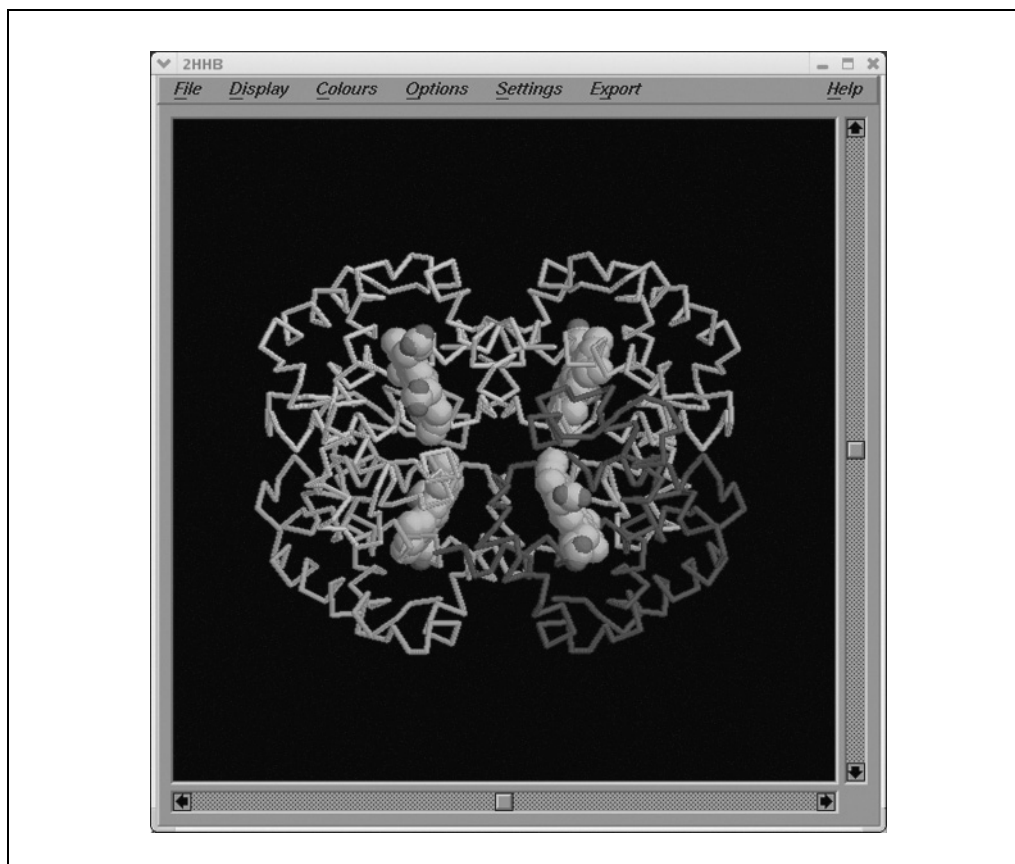


Figure 5.4.6 A quick overview representation of hemoglobin. For the color version of this figure go to <http://www.currentprotocols.com>.

3. Rotate the display and note that this representation quickly makes it possible to see that: (1) hemoglobin is composed of four similar chains with lots of alpha helices and (2) there are four hemes that are sandwiched between alpha helices.

Finding key residues

When looking for a particular amino acid, it is possible to examine a wireframe representation. This tends to be rather confusing, however, and it may be difficult to find the desired amino acid among the many surrounding ones. By using a simple combination of selection and representation commands, this process may be facilitated. The following example shows an easy way to find the histidine residues in hemoglobin that interact with the iron ions, without the need to go to the literature to find the residue number.

4. From the overview representation presented above, type, in the Command Line window:

```
RasMol> select his
```

This selects all histidines.

```
RasMol> cpk
```

This draws the histidines with spheres.

The display should look like Figure 5.4.7.

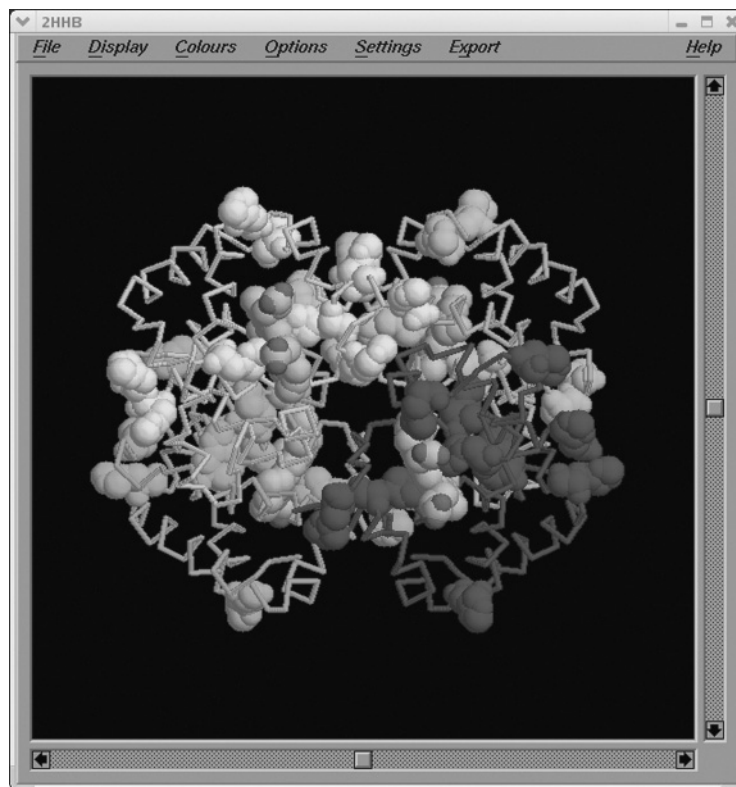


Figure 5.4.7 All histidines in hemoglobin are shown with spacefilling spheres. For the color version of this figure go to <http://www.currentprotocols.com>.

5. At this point it is fairly simple to zoom in on one of the hemes, as in Figure 5.4.8, and click on one of the histidine atoms to find the residue number. This is a bit tricky with hemoglobin, since it has histidines on both sides of the heme. However, by looking closely, it is possible to see that one histidine is coordinated directly to the iron. In this case, the view is centered on HIS92 in chain D.
6. Clean up the picture by typing the following series of commands in the Command Line window:

```
RasMol> cpk off
```

This turns off the spheres on all the histidines.

```
RasMol> select HIS92:D
```

This selects just histidine 92 in chain D.

```
RasMol> wireframe 100
```

This draws a thick wireframe on this histidine.

```
RasMol> color cpk
```

This colors the histidine by atom type.

This should give a display like the one in Figure 5.4.9.

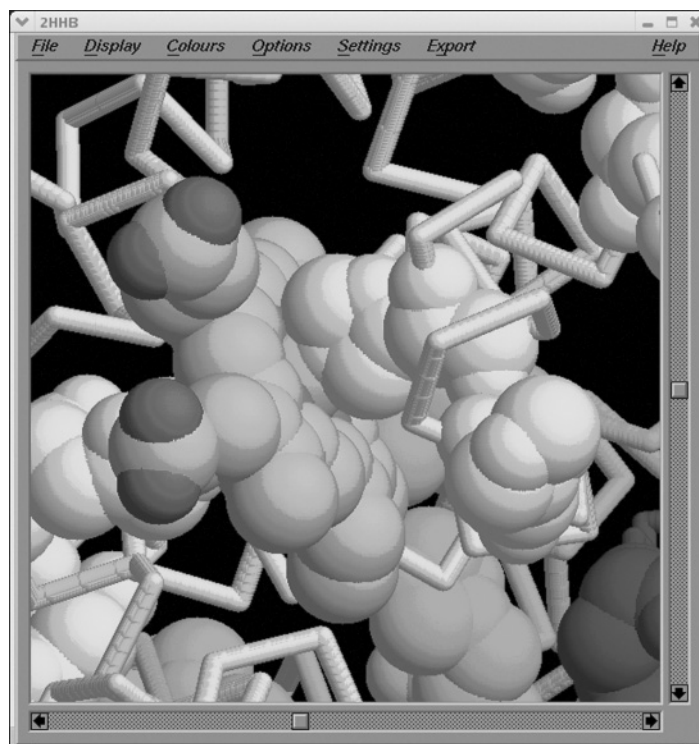


Figure 5.4.8 Zooming in on one heme group, it is easy to locate histidines on either side of the iron ion. The one on the right is histidine 92, which coordinates with the iron ion. For the color version of this figure go to <http://www.currentprotocols.com>.

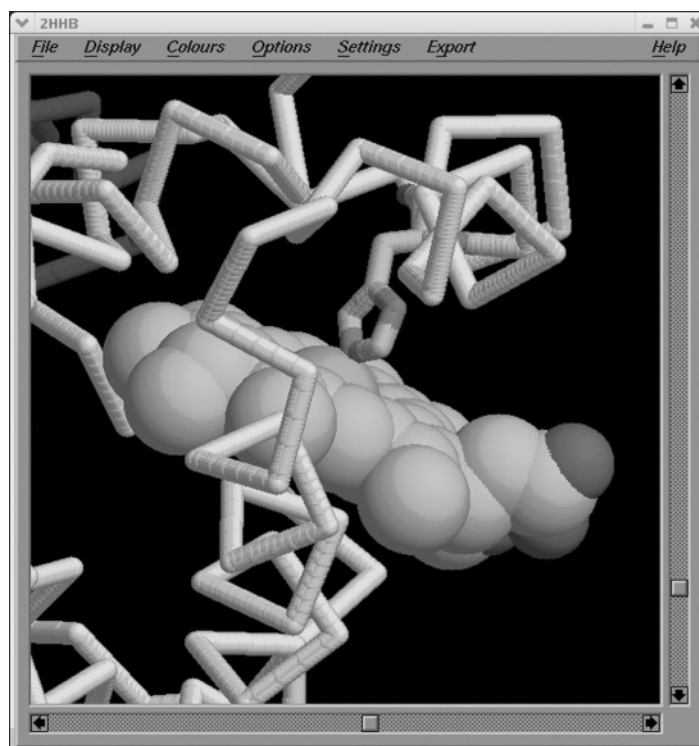


Figure 5.4.9 Histidine 92 is displayed with a thick wireframe representation, colored by atom type. For the color version of this figure go to <http://www.currentprotocols.com>.

7. Notice that this display is a bit messy because the backbone atoms are included for the histidine. This can be cleaned up with:

```
RasMol> wireframe off
```

This turns off the wireframe on histidine 92.

```
RasMol> select HIS92:D and (sidechain or alpha)
```

This selects only the sidechain atoms and the alpha carbon atom in histidine 92.

```
RasMol> wireframe 100
```

This draws the histidine 92 sidechain with a thick wireframe.

VIEWING THE APPROPRIATE BIOLOGICAL UNIT

Coordinate files from the PDB are full of surprises. This is sometimes a source of delight, but often a source of frustration. A major challenge when examining a structure is determining whether it includes an appropriate biological unit. The biological unit is defined as the physiologically relevant state of the molecule, such as a complex of four chains in hemoglobin or an entire icosahedral structure in a viral capsid. Unfortunately, the coordinate sets obtained from the PDB, since they are subject to the methodology used in the structure determination, do not always include exactly one biological unit. The challenge is to generate a file that includes coordinates for the entire biological unit. This protocol describes how to view the appropriate biological unit using RasMol.

Necessary Resources

Hardware

RasMol runs on a variety of computer hardware, including personal computers.

Software

Operating system: RasMol runs under Microsoft Windows and Apple Macintosh OS 7.0 or higher (including Mac OS X). It may also be run on workstations under Unix, Linux, or VMS.

RasMol: Binary versions of RasMol are available on the WWW at: <http://www.bernstein-plus-sons.com/software/rasmol/>. Downloading and installation instructions are given in Support Protocol 1.

Files

Coordinate files are read in a variety of formats, including PDB, Mol2, CHARMm, and mmCIF. The program deals gracefully with a number of variations of these files, including files containing coordinates for multiple conformers or multiple models. In this example, coordinates for hemoglobin (2HHB.pdb) obtained from the Protein Data Bank (PDB; UNIT 1.9) are used; instructions for downloading the PDB coordinate file are given in Support Protocol 2.

BASIC PROTOCOL 2

1. Three different problems with biological units may be encountered as one goes to the Protein Data Bank for coordinates. First, the coordinate file may include only a portion of the physiologically active complex. The examples in this unit have been using the deoxygenated form of hemoglobin so far, which as four protein chains. However, an overview picture of PDB entry 1hho, the oxygenated form of hemoglobin, will look like Figure 5.4.10.
2. Notice that there are only two chains in the file, even though it is known that hemoglobin is active with four chains. This is due to the details of the crystallographic experiment, where the two halves of the protein are crystallographically identical in the structure, so the researchers only report coordinates for half. Fortunately, the need for appropriate biological units has become clear, and the PDB has a facility for downloading coordinate sets with the presumed biological unit. These may be found at the bottom of the Download/Display File page for the structure, as shown in Figure 5.4.11.

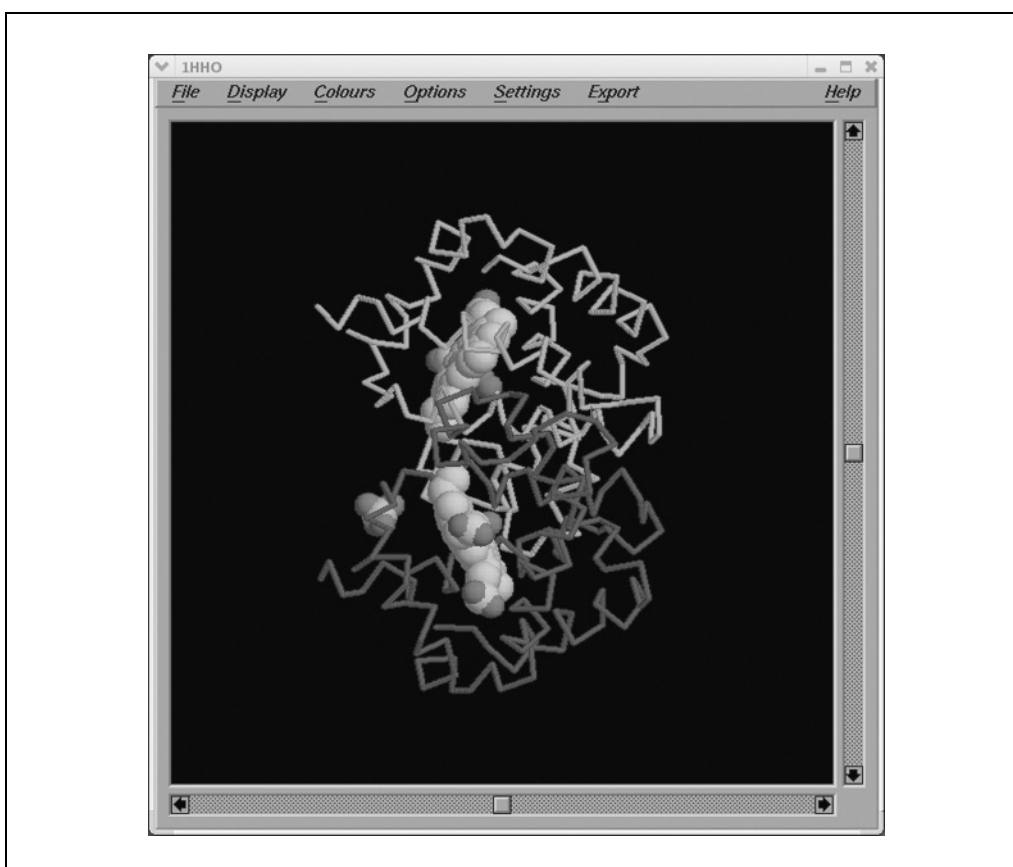


Figure 5.4.10 Overview representation of the coordinate file for oxyhemoglobin in PDB entry 1hho. For the color version of this figure go to <http://www.currentprotocols.com>.

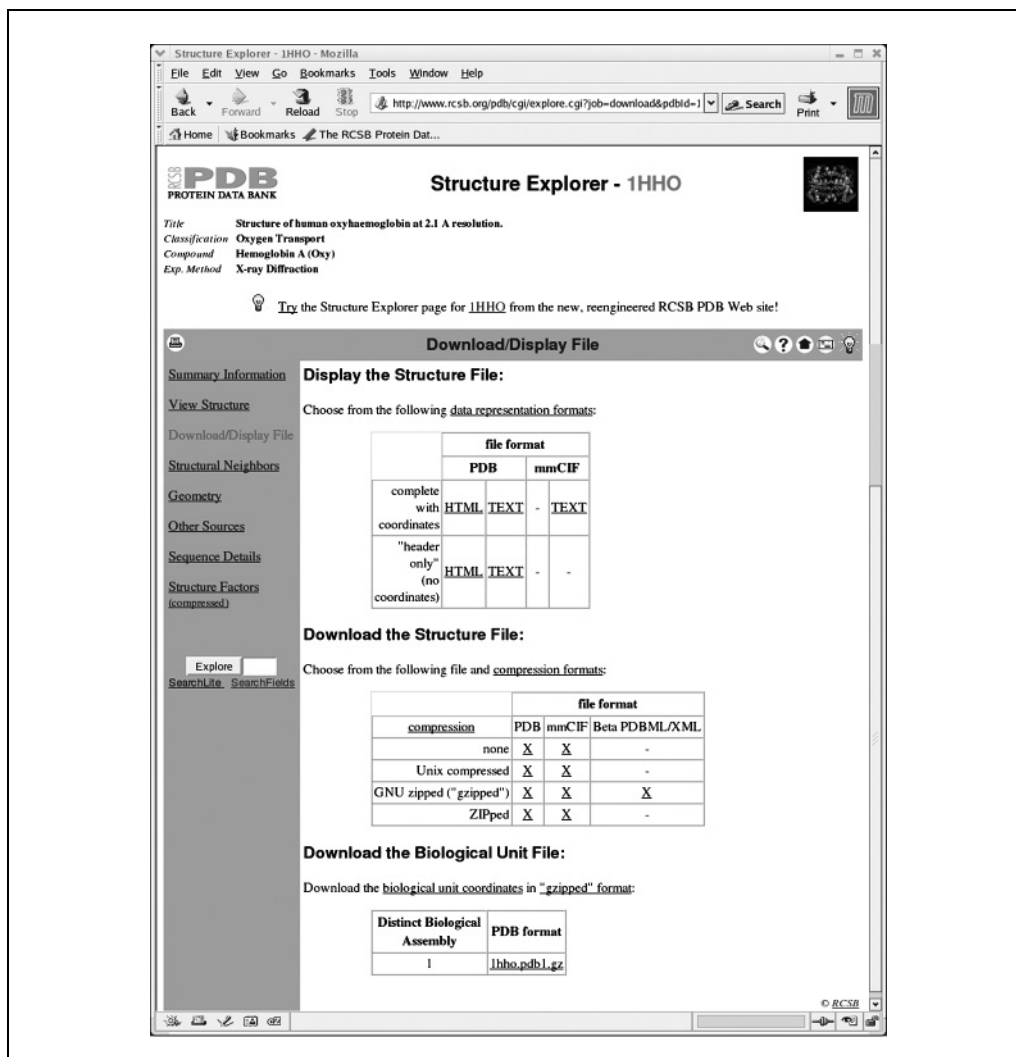


Figure 5.4.11 The Download/Display File page for oxyhemoglobin at the PDB. The link at the bottom of the page allows access to coordinates of the biological unit.

3. The opposite problem also occurs in other structure files. In these cases, there are multiple biological units in the coordinate file, again due to the details of symmetry and packing of molecules in the crystal. For instance, PDB entry 1hbs includes eight chains, forming two complete hemoglobin tetramers, as shown in Figure 5.4.12. In this case, however, the multiple structure is interesting, since it shows the presumed stacking of this sickle-cell hemoglobin. To show only the biological unit—i.e., the tetramer—the chain identifiers can be used to blank out one of the hemoglobin tetramers. Alternatively, it is often easiest to edit the coordinates directly, using a text editor to remove the unwanted chains.
4. Another problem occurs when looking for proteins that are large or flexible. In these cases, the researchers may have trimmed off flexible portions or cut the protein into pieces for individual study. The example shown in Figure 5.4.13 is ATP synthase, which has been solved in different parts. These two pieces were taken from PDB entries 1c17 and 1e79. There is no quick solution to this problem, unfortunately. Careful study of the published reports is necessary to ensure that the functionally relevant portion of the molecule is being displayed.

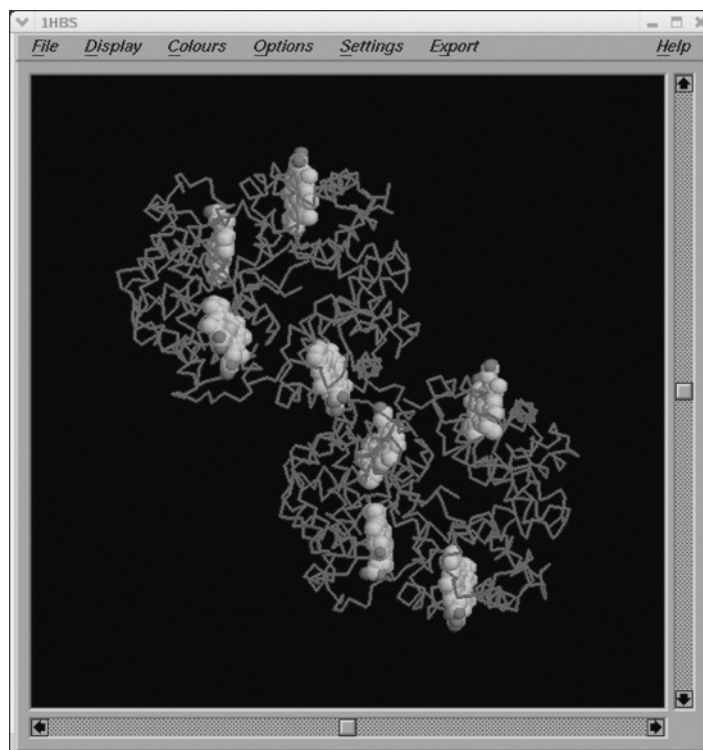


Figure 5.4.12 Overview representation of sickle-cell hemoglobin from PDB entry 1hbs. For the color version of this figure go to <http://www.currentprotocols.com>.

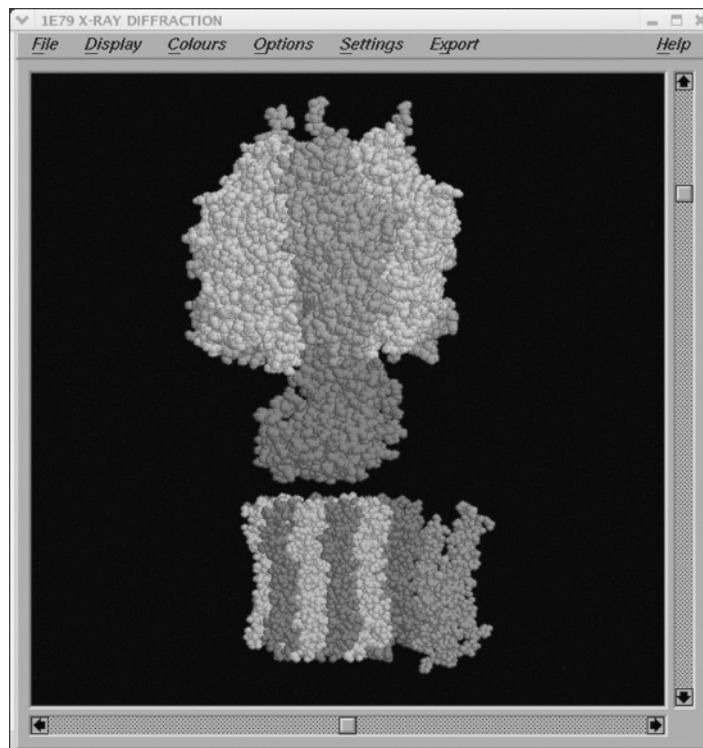


Figure 5.4.13 ATP synthase in a spacefilling representation. For the color version of this figure go to <http://www.currentprotocols.com>.

CUSTOMIZING A RasMol SESSION

When beginning to use a new molecular graphics program, it is common practice to use the default parameters during the learning process. However, these default settings are only guidelines, and many simple modifications can improve the utility of the program for different applications. The important thing is to understand the goal of the representation when beginning. For instance, one type of display is needed to understand the effect of a point mutation in hemoglobin, and a different display is needed to show the allosteric changes between oxy and deoxy forms. Much of the artistry, and the fun, of molecular graphics begins when displays are customized for specific applications, as described in this protocol.

To develop sophisticated displays, it is useful to use the scripting function of RasMol. This makes it possible to type all of the commands in a separate text file and then read them into RasMol. The command is `RasMol> script file.txt`, where `file.txt` is the name of the script file.

Necessary Resources

Hardware

RasMol runs on a variety of computer hardware, including personal computers.

Software

Operating system: RasMol runs under Microsoft Windows and Apple Macintosh OS 7.0 or higher (including Mac OS X). It may also be run on workstations under Unix, Linux, or VMS.

RasMol: Binary versions of RasMol are available on the WWW at: <http://www.bernstein-plus-sons.com/software/rasmol/>. Downloading and installation instructions are given in Support Protocol 1.

Files

Coordinate files are read in a variety of formats, including PDB, Mol2, CHARMm, and mmCIF. The program deals gracefully with a number of variations of these files, including files containing coordinates for multiple conformers or multiple models. In this example, coordinates for hemoglobin (2HHB.pdb) obtained from the Protein Data Bank (PDB; UNIT 1.9) are used; instructions for downloading the PDB coordinate file are given in Support Protocol 2.

Color management

RasMol recognizes a number of common colors with commands such as `color red` or `color blue`. These tend to be saturated colors, however, which rapidly become confusing in complex pictures. For instance, the pictures of hemoglobin shown in the figures illustrating the previous protocols use the default chain colors, which are all bright primary and secondary colors. Saturated colors compete with each other on the screen and often confuse the perception of the relative depth of different portions of the molecule. It is possible to use custom colors to design a picture that minimizes these artifacts and focuses more attention on the functional details. Pastel colors are often easier to read, and they do not compete with each other in the display. RasMol does not contain a graphical color browser, but it does allow the user to design custom colors.

1. Restart RasMol with the file `2hbb`, and in the Command Line window type the following series of commands:

```
RasMol> select protein or ligand
```

```
RasMol> cpk
```

```
RasMol> select *:A
```

This selects all atoms in chain A.

```
RasMol> color [100,100,255]
```

This colors the chain light blue.

```
RasMol> select *:C
```

This selects chain C.

```
RasMol> color [100,150,255]
```

This colors the chain blue-green.

```
RasMol> select *:B
```

This selects chain B.

```
RasMol> color [100,255,100]
```

This colors the chain green.

```
RasMol> select *:D
```

This selects chain D.

```
RasMol> color [50,255,150]
```

This colors the chain aqua.

```
RasMol> select ligand
```

This selects the heme groups.

```
RasMol> color [255,100,100]
```

This colors the hemes pink.

The display should look like Figure 5.4.14. Notice how the color differences are still apparent, but they do not distract from the inter-relationship of the subunits within the entire structure.

2. To get an impression of the limitations of saturated colors, now type:

```
RasMol> select ligand
```

```
RasMol> color red
```

3. Notice how the saturated red causes confusion between the heme group and the surrounding protein chain. The impression of the heme being buried in a pocket is not as clear. However, if the goal is to focus all attention on the hemes, this bright red might be the best choice.
4. Use of the `color` command takes some practice in order to come up with the desired color. The values in the brackets are the intensity of red, green, and blue, with ranges from 0 to 255. The easiest way to start is to begin with a saturated color, and then modify it to give the desired color. In most cases, it will take a few experiments to get the proper color. Here is an example when looking for a peach color. First type:

```
RasMol> select ligand
```

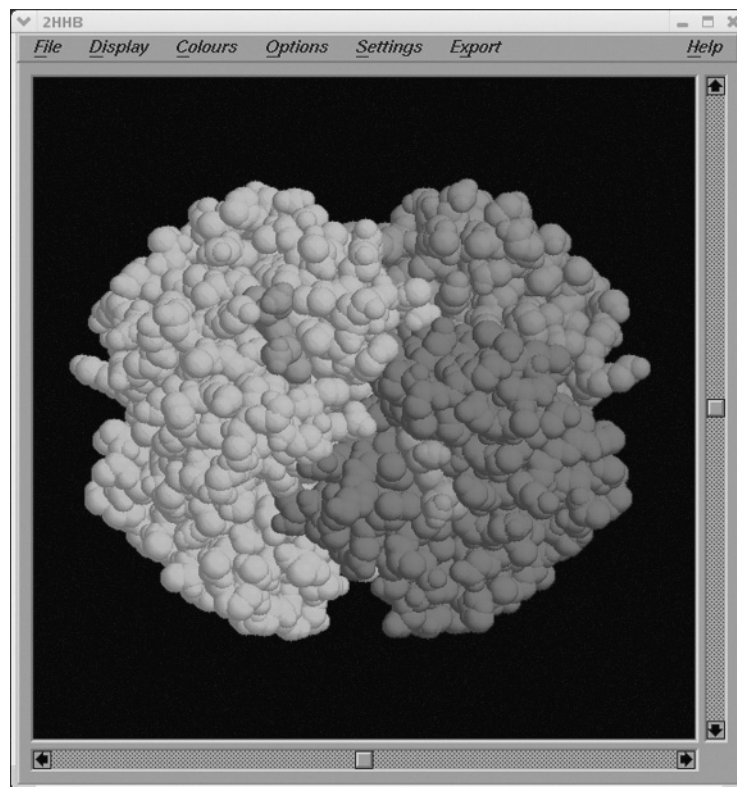


Figure 5.4.14 An alternate coloring scheme for hemoglobin. For the color version of this figure go to <http://www.currentprotocols.com>.

then modify it using the following steps:

- a. Start with saturated red:

```
RasMol> color [255,0,0]
```

- b. Try a little more green to get bright orange:

```
RasMol> color [255,100,0]
```

- c. Now raise everything by 50 to get a lighter color (except red, because it is already at the maximum of 255):

```
RasMol> color [255,150,50]
```

- d. Raise by 50 more to get the pastel peach:

```
RasMol> color [255,200,100]
```

Combinations of representations

The best picture for most applications will be composed of a number of different representations. For instance, the overview representation shown above uses backbones for the proteins and spacefilling representations for the ligands. The backbones are simple, showing at a glance the whole structure and the relationships between the protein chains. The ligands, however, are small, so the bulky spacefilling representation is used to make sure that they stand out in a complex structure. Most molecular graphics programs give considerable flexibility in the modification of these representations. For instance, it is possible to vary the diameter of the cylinders used in wireframe

representations and add small balls at the atom positions, to help distinguish different parts of the structure. One way to improve the clarity of a given picture is to stick to a common representation for each part of the structure. The example shown in Figure 5.4.9 is an example. The backbone representation used for the protein and the wireframe used for the histidine have a similar look, so the viewer automatically treats them as part of the same structure, even though the coloring scheme is different between the backbone and the sidechain. The heme is shown in spacefilling, so it is distinguished as a different molecule.

5. To see how this works, restart RasMol with the file 2hhb, and type:

```
RasMol> select all  
RasMol> wireframe off
```

This turns off the default representation.

```
RasMol> select protein  
RasMol> backbone 100
```

This uses a thick protein backbone.

```
RasMol> color [100,150,255]
```

This colors the protein backbone blue-green.

```
RasMol> select ligand  
RasMol> cpk
```

This uses spheres for the heme.

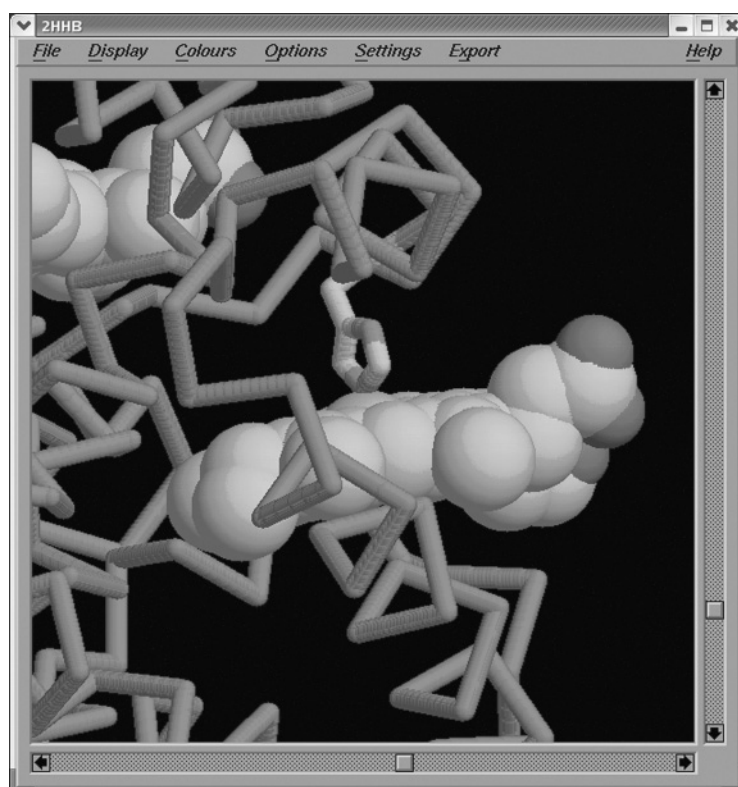


Figure 5.4.15 A close-up image of the histidine-iron interaction in hemoglobin. For the color version of this figure go to <http://www.currentprotocols.com>.

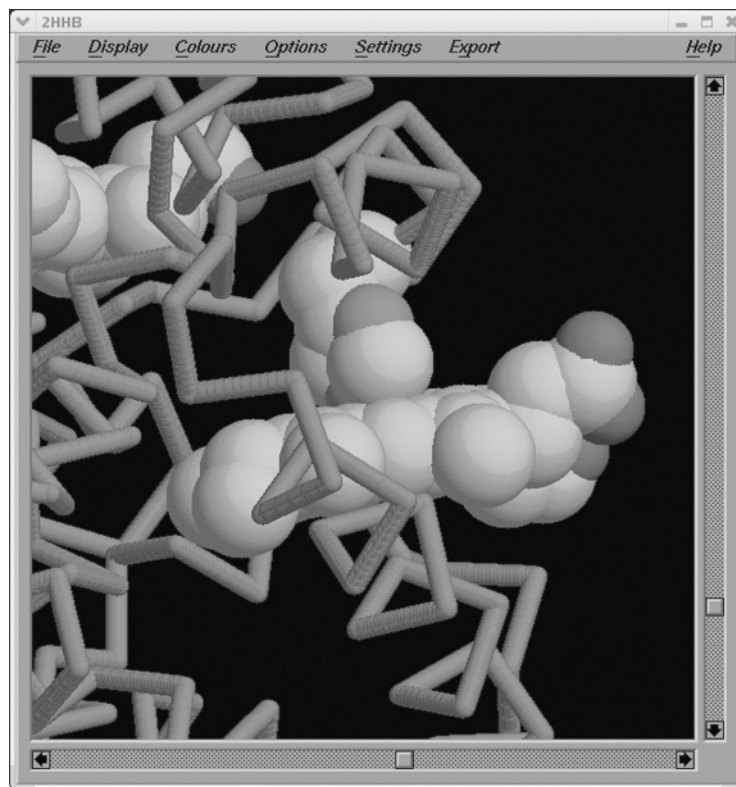


Figure 5.4.16 An alternate close-up image of the histidine-iron interaction in hemoglobin. For the color version of this figure go to <http://www.currentprotocols.com>.

```
RasMol> select HIS92:D and (sidechain or alpha)
RasMol> wireframe 100
```

This uses wireframe for the histidine.

```
RasMol> color cpk
```

This colors the histidine by atom type.

6. Rotate and scale the display to find a satisfactory view of the interaction, like that in Figure 5.4.15.

7. Type the following command:

```
RasMol> cpk
```

This will use a spacefilling representation for the histidine, as in Figure 5.4.16. Notice that the picture is more confusing now, and it is difficult to tell if the histidine is part of the protein or part of the heme. By mixing different representations, one always runs the risk of creating this type of confusion.

GUIDELINES FOR UNDERSTANDING THE RESULTS

To create effective molecular graphics requires a combination of scientific background and aesthetic judgement. When approaching a new project, it is first necessary to define what needs to be shown, and then develop a representation that clearly shows it. Two guidelines will assist in this process.

Define the Medium and the Audience

Before sitting down at the computer, it is important to understand the goals of the graphics session. For instance, at the beginning of a project the goal may be to display an entirely new structure and do some exploration. Alternatively, the goal may be to create a figure for journal publication that shows the specifics of binding of a ligand within an enzyme active site. These two goals will each require entirely different approaches to the subject, and may be best served with two entirely different molecular graphics programs.

Parameters to define when beginning a project include:

The medium of presentation. Interactive display will allow the use of very complex representations, whereas print media require simpler representations that will be comprehensible in a still image.

The audience. Images created for molecular biologists typically can be far more complex than images created for the lay audience or for researchers in other fields. Researchers are often willing to spend more time with an image to ferret out all of the details.

Set Achievable Goals

When designing a representation for a given goal, it is important to set achievable goals. It is rarely possible to show many concepts in a single figure. Instead, it is often best to pick one concept and create a representation that best serves that goal. For instance, the overview representation given above is only good for one purpose: to give an overview of the protein fold and the location of ligand-binding sites. If the details of the ligand-binding site were added, perhaps by adding all of the sidechains that interact with the ligand, the representation would suffer. The binding site would become too complex and would distract from the global features of the protein, and the details of the active site would be so small that they would not be comprehensible. The better approach is to split this into two figures: an overview to show the context and a close-up to show the details.

COMMENTARY

Background Information

A decade ago, molecular graphics was the domain of experts in computer graphics, but today a wide variety of molecular graphics programs are available, allowing researchers, students, and educators to create their own molecular illustrations. Since molecules are themselves smaller than the wavelength of light, a metaphor must be employed to create a model that captures some properties of the molecule in visual form. Several of these metaphors have had lasting success: bond diagrams to show the covalent geometry of the molecule, spacefilling diagrams to show the shape and form of the molecule, and backbone representations to show the topology and folding of a macromolecular chain. Most molecular graphics programs allow the user to create an image of a molecule using a combination of these rep-

resentations, making it possible to tailor the image to one's own application.

Critical Parameters and Troubleshooting

Most computer graphics programs contain hundreds of user-controlled parameters for selecting and displaying different portions of molecules. These programs also provide default values for these parameters, so that an initial image may be generated rapidly. These defaults should provide a guide, but not a limitation, to the creative process.

Default parameters are chosen by the programmer for a good reason: they are the best guess for what the user will most often need. In many cases, they will also define a representation that corresponds to what a viewer will expect to see. For instance, most programs

provide a familiar atomic coloring scheme, using black/gray for carbon, red for oxygen, blue for nitrogen, and so on. Before changing this default coloring scheme, it is worth thinking about how the picture will be viewed. Many of the defaults provided by graphics programs are designed to create familiar images with chemical features that are recognizable at a glance. For instance, if the color of all of the oxygens is changed to yellow, most viewers will automatically assume that they are sulfurs, potentially causing confusion. The radii of spacefilling representations are another example of defaults that should be respected, since they are designed to show a particular physical characteristic of the molecule.

That being said, default parameters are only guidelines, and should be modified to suit the current goal. Color, in particular, is a powerful tool for directing attention to key features, and default parameters are rarely able to draw attention to exactly the feature that needs to be highlighted. The width of cylinders in backbone and bond diagrams provide another effective avenue for customizing a representation.

Suggestions for Further Analysis

Fortunately, when beginning to explore the capabilities and possibilities of molecular graphics, there is a rich tradition to build upon. As with other artistic techniques, a good way to choose an approach for a particular application is by example. A number of reviews are available (Goodsell, 2003, 2005; Olson and Goodsell, 1992a,b; Richardson, 1992) to provide an overview of approaches and tech-

niques. It is also highly instructive to browse through a few issues of *Science*, *Nature*, or *Structure*, and look for figures that are particularly effective. This is a good way to preview the capabilities of different programs before investing the necessary time to master them. But most important, have fun and explore the many possibilities while developing an individual graphical style.

Literature Cited

- Goodsell, D.S. 2003. Looking at molecules: An essay on art and science. *ChemBioChem* 4:1293-1298.
- Goodsell, D.S. 2005. Visual methods from atoms to cells. *Structure* 13:347-354.
- Olson, A.J. and Goodsell, D.S. 1992a. Macromolecular graphics. *Curr. Opin. Struct. Biol.* 2:193-201.
- Olson, A.J. and Goodsell, D.S. 1992b. Visualizing biological molecules. *Sci. Am.* 267:76-81.
- Richardson, J.S. 1992. Looking at proteins: Representations, folding, packing, and design. *Bio-phys. J.* 63:1186-1209.

Internet Resources

<http://www.rcsb.org/pdb>

Web site for the Protein Data Bank (PDB).

<http://www.rcsb.org/pdb/software-list.html>

Contains links to many molecular graphics programs and provides access to macromolecular coordinates.

Contributed by David S. Goodsell
The Scripps Research Institute
La Jolla, California

Using Dali for Structural Comparison of Proteins

Dali (distance matrix alignment) is a tool for both pairwise structure comparison and structure database searching. It is equipped with a Web interface to easily view the results, multiple alignments, and three-dimensional (3D) superimpositions of structures. The method is fully automated and very sensitively identifies common structural cores and structural resemblances. Dali uses 3D Cartesian coordinates of C α atoms of each protein in order to calculate residue-residue distance matrices. A similarity score for these sets is defined as a weighted sum of equivalent intramolecular distances, resulting in a scored list of all important structural alignments. This method allows for any length of gaps in the sequence (i.e., insertions or deletions) and detects similarities involving geometrical distortions.

Dali is easily accessible through Web servers, and Table 5.5.1 outlines the relationships of Dali resources. The DaliLite server can be used to compare two known structures to each other and visualize their superimposition (Basic Protocol 1). This server requires two sets of atomic coordinates in PDB format as input. The comparison is usually quite fast, and results should be returned after about one minute. A search against all known structures takes much longer and can be performed using the DALI Server (Basic Protocol 2). This server is routinely used by protein crystallographers to compare a newly solved structure to known structures in the database in order to detect possible evolutionary relationships. The structure neighbors of proteins already in the PDB (Protein Data Bank) can be found in the Dali database. Its Web interface allows browsing of the hierarchical classification of protein structures based on all-against-all comparisons of known structures (Basic Protocol 3: Dali database).

Table 5.5.1 Overview of Dali Resources and Their Relations

	DaliLite	Dali server	Dali database	ADDA database
Input	Two (lists of) PDB structures	One PDB structure	All PDB structures	All known protein sequences
Steps	Pairwise structure comparison	Database search using cascaded algorithms	Remove redundancy All-against-all structure comparison Domain decomposition Clustering	Remove redundancy All-against-all sequence comparison Domain decomposition Clustering
Output	Structure neighbors of query	Structure neighbors of query	Protein fold classification	Protein family classification
Protocol	Basic Protocol 1 Alternate Protocols 1 and 2 Support Protocol	Basic Protocol 2	Basic Protocol 3	Linked to Dali database

When it is necessary to query many structures, it may be convenient to download the DaliLite stand-alone program package. This package uses the same comparison algorithms as the Dali Web servers but can be run locally on Linux-based computers (see Alternate Protocol 1, Alternate Protocol 2, and the Support Protocol).

USING THE INTERACTIVE DaliLite SERVER FOR PAIRWISE COMPARISONS

This interactive Web server provides a quick, convenient means for checking the structural alignment of two known protein structures and for visualizing their structural superimposition. Only the PDB identifiers of the structures are required. It is also possible to upload user-specific structures. A fast server can be accessed at <http://www.ebi.ac.uk/DaliLite>.

Necessary Resources

Hardware

Computer connected to the Internet

Software

Internet browser (e.g., Internet Explorer, <http://www.microsoft.com>; Netscape, <http://browser.netscape.com>; or Firefox, <http://www.mozilla.org/firefox>)
RasMol (UNIT 5.4; downloadable from <http://www.bernstein-plus-sons.com/software/rasmol/>) or other PDB viewer

Files

User-specific PDB files, optional

1. Go to <http://www.ebi.ac.uk/DaliLite>. The submission page for pairwise comparison of protein structures is shown in Figure 5.5.1.

Figure 5.5.1 Submission page of the DaliLite server.

2. Input First and Second Structures in the submission page (Fig. 5.5.1) as PDB entry codes (for known structures) or upload user-specific coordinate files in PDB (UNIT 1.9) format. For example, to compare the structures of 1qku (estrogen nuclear receptor, ligand binding domain) and 1k4w (orphan nuclear receptor, ROR beta ligand-binding domain), enter the PDB identifiers in the “PDB entry code” boxes as shown in Figure 5.5.1, or enter the .pdb filenames in the lower row of input boxes in the Figure 5.5.1.

Searches for the PDB entry codes of known structures for a query protein can be performed using Entrez at NCBI (<http://www.ncbi.nlm.nih.gov>), SRS (<http://srs.ebi.ac.uk>), and other similar database cross-linking resources.

For a structure file containing a number of different chains, a specific chain can be selected in the submission page. If no chain is specified, structural comparisons will be performed on every chain in the structure file, and the return of results will take much longer.

Size limits for the comparison are between 30 and 1000 amino acid residues per chain.

3. Click on the Run DaliLite button. The summary page for the results of a structure comparison appears; the top part of the page is shown Figure 5.5.2.

The page (Fig. 5.5.2) includes the following information:

Z-Score: The Z-Score is a measure of quality of the alignment—the higher, the better. As a general rule, Z-scores above 8 yield very good structural superimpositions, Z-scores between 2 and 8 indicate topological similarities, and Z-scores below 2 are not significant.

Aligned Residues: The number of aligned residues is the number of structurally equivalent residue pairs.

RMSD: The root-mean-square deviation (RMSD) is a measure of the average deviation in distance between aligned alpha carbons in structural superimposition. Long alignments, e.g., over 100 aligned residues with RMSD below 3 Å, indicate similar folds.

Sequence identity: It is generally assumed that if sequences of two chains share over 40% identity, then they are unambiguously homologous and structurally very similar. However, distantly related proteins may share very low sequence identity but still be structurally similar.

For each chain in the query structure, a table is presented showing significant hits against each chain of the subject structure. Note that the first structure is named “mol1”, the second structure is named “mol2”, chain A of the first structure is “mol1A”, and so on. Suboptimal alignments are reported; the highest scoring alignment per any pair of chains is highlighted by light blue background.

4. To access information in the table for Results of Structure Comparison about structural alignments (including secondary structure information) between the indicated chains, click the “click here” link under the Structural Alignment category to generate the alignment shown in Figure 5.5.3.
5. To generate a coordinates file of the superimposed C α traces for the indicated chains, viewable in RasMol (UNIT 5.4) or other PDB structure viewers, click the CA_1.pdb link under Superimposed C-alpha Traces. In the example, the C α trace shown in Figure 5.5.4 is generated.

Only the C α coordinates are transmitted; therefore use the backbone display in RasMol! Note that in the coordinates sent to RasMol the first structure chain (from mol1) is renamed Q, and the second structure chain (from mol2) is renamed S.

6. To view the full superimposition, either open both files under the heading “PDB Files: mol2 is rotated/translated to mol1 position” in the PDB viewer, or concatenate the two files and view the resulting file. The second option preserves ligands that might have been co-crystallized with the protein as well as showing quaternary structure interactions.

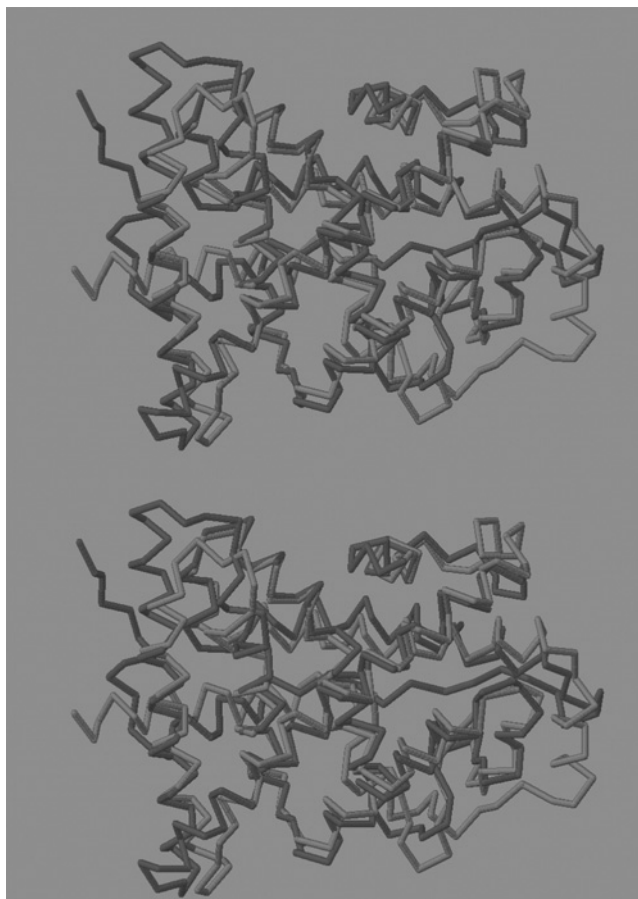


Figure 5.5.4 Superimposition of the two protein chains in RasMol (stereo view) obtained by clicking on the “Superimposed C-alpha traces” link on view shown in Figure 5.5.2. The query structure (mol1) is blue, and the second structure (mol2) is red. For the color version of this figure go to <http://www.currentprotocols.com>.

In the example in Figure 5.5.2 the link to the first structure file (unchanged) is called mol1_original.pdb. The second structure file with all ATOM coordinates of the indicated chain rotated/translated to match the first structure is called mol2_1.pdb.

Note that only the indicated chains are superimposed (e.g., mol1A with mol2B). However, since any other chains will still be contained in the structure files, it may be desirable to remove unwanted chains using a text editor before viewing the structures.

7. To view the files for Rotation/translation matrices for each alignment, Listing of structurally equivalent residue ranges, and View the log (indicating all the steps taken by the DaliLite application), click on the hyperlinks under the “Additional data” header.

These files are included for completeness but are not important to most users.

8. Check the data under the Inputs header at the bottom of the results page for a summary of the two inputs, including header information and a report of the chains found within each structure file.

If these data are not as expected, it is apparent that file upload (rather than the program itself) may have failed for one reason or another.

SEARCHING FOR STRUCTURAL NEIGHBORS USING THE Dali E-MAIL SERVER

The Dali server is an easy-to-use network service for comparing protein structures. It is routinely used by structural biologists to compare a newly solved structure against previously known structures. In favorable cases, comparing 3D structures may reveal biologically interesting similarities that are not detectable by comparing sequences. Submitting the coordinates of a query protein structure to Dali compares them to those in the Protein Data Bank, and a multiple alignment of structural neighbors is e-mailed back. Structural neighbors of a protein already in the Protein Data Bank can be found in the Dali database (Basic Protocol 3). The Dali server (<http://www.ebi.ac.uk/dali>) is hosted by the European Bioinformatics Institute (EBI). Structure submission can be made either interactively or by e-mail. E-mail submission may be more convenient for larger sets of queries.

Necessary Resources

Hardware

Computer connected to the Internet

Software

Internet browser (e.g., Internet Explorer, <http://www.microsoft.com>; Netscape, <http://browser.netscape.com>; or Firefox, <http://www.mozilla.org/firefox>)
E-mail account

Files

Atomic coordinates of protein structure in PDB format

To submit coordinates interactively

- 1a. Go to <http://www.ebi.ac.uk/Interactive.html>. The submission page is shown in Figure 5.5.5
- 2a. Click on the “3D structure x PDB database” link below “Database search” to access the “Database search form” shown in Figure 5.5.6. Type in the e-mail address to which results are to be sent, ignore the password box, and upload the coordinate file. Click on the “Submit query” button.

The results will be sent to the e-mail address provided on the submission page. Type carefully.

To submit coordinates by e-mail

- 1b. Send an e-mail message containing the PDB coordinates in plain text to dali@ebi.ac.uk.

The submission will fail unless the message is plain text. Encoded messages (e.g., MIME or BinHex) are rejected by the server.

- 2b. An e-mail with the results may be expected within a few days of submission. In case of longer delays, notify dali-help@ebi.ac.uk.

The comparison is carried out against a representative subset of PDB structures. The set is constructed so that the sequence identity between any two chains in the set should be less than 25%. Proteins with higher sequence identity usually have very similar folds. A typical summary of structural neighbors is shown in Figure 5.5.10. See Basic Protocol 3 for a description of this.

3. Use the DaliLite server for pairwise comparison (Basic Protocol 1) to visualize interesting pairs of structures.

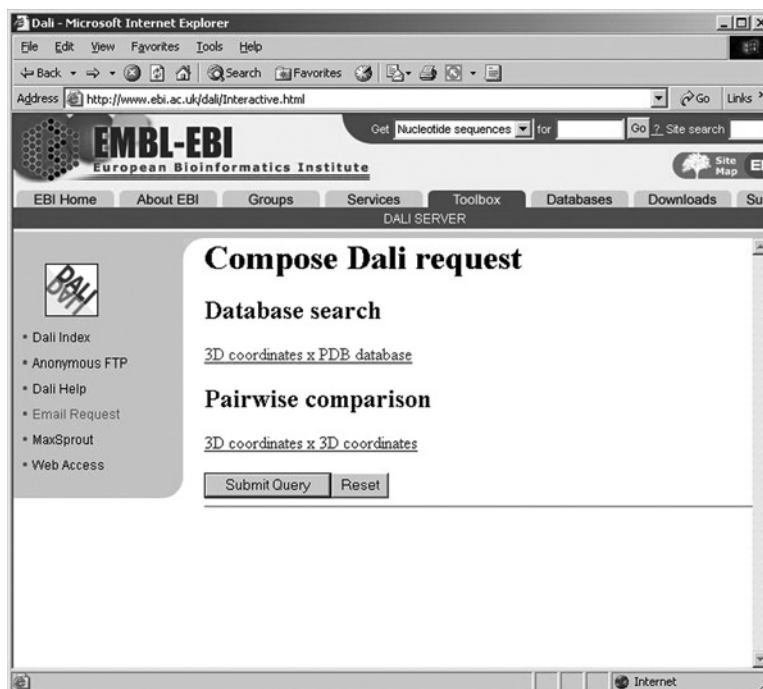


Figure 5.5.5 Interactive submission menu of the Dali server.

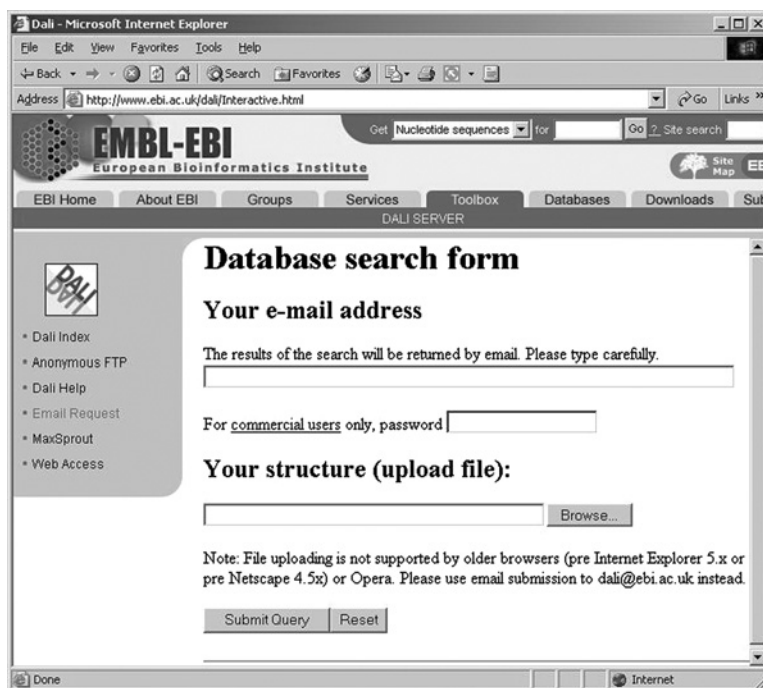


Figure 5.5.6 Submission page proper of the Dali server.

USING THE Dali DATABASE TO INVESTIGATE FAMILIAL RELATIONS AMONG THE UNIVERSE OF PROTEIN FOLDS

The Dali database is based on exhaustive all-against-all 3D structure comparison of protein structures currently in the Protein Data Bank (PDB). The classification and alignments are automatically maintained and continuously updated using the Dali search engine. The database currently contains 10,562 representative structures (May 2006). This protocol describes how to search for familial relationships among the known set of protein folds.

Necessary Resources

Hardware

Computer connected to the Internet

Software

Internet browser (e.g., Internet Explorer, <http://www.microsoft.com>; Netscape, <http://browser.netscape.com>; or Firefox, <http://www.mozilla.org/firefox>)
RasMol (UNIT 5.4; downloadable from <http://www.bernstein-plus-sons.com/software/rasmol>) or other PDB viewer

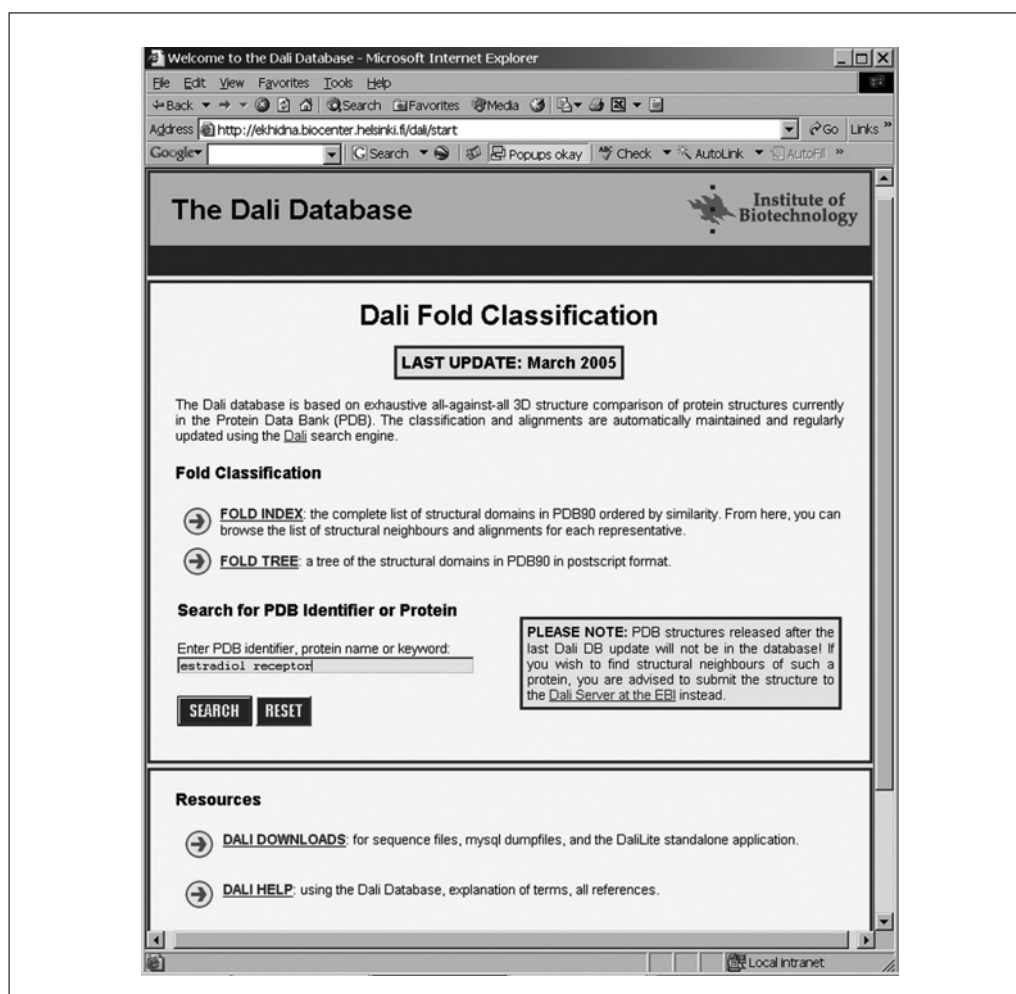


Figure 5.5.7 Home page of the Dali database. The user has typed `estradiol receptor` in the query box.

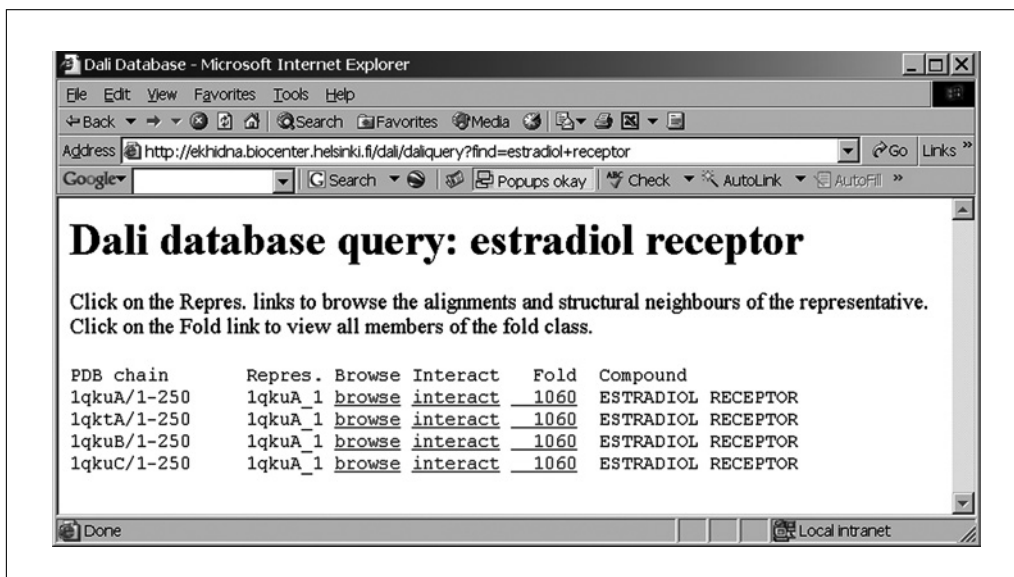


Figure 5.5.8 The result of the query for “estradiol receptor” structures.

Browse the Dali database

1. Go to the Dali database at <http://www.bioinfo.biocenter.helsinki.fi/dali/start>. The home page is shown in Figure 5.5.7.

The set of representative structures is called PDB90, and it contains all polypeptide chains from the PDB with less than 90% sequence identity to each other. The representative structures are decomposed into 14,020 domains. Hierarchical clustering reveals 3,107 fold types. Fold types are defined as clusters of structural neighbors in fold space with average pairwise Dali Z-scores above 2. The threshold has been chosen empirically and groups together structures that have topological similarity. Higher Z-scores correspond to structures that agree more closely in architectural detail. The Fold Index lists all chains in PDB90 ordered by structural similarity. The order is that of a dendrogram derived in the hierarchical clustering. Fold types are indexed. A “heavier” branch with more members is listed above a branch with fewer members. Domains that are structural neighbors are found next to each other. Fold types with similar structural motifs are also found next to each other.

2. Enter into the fold classification from the FOLD INDEX or enter a PDB identifier or text term (protein name or keyword) that occurs in the COMPND records of the PDB entries into the text box under Search for PDB Identifier or Protein (Fig. 5.5.7).

More sophisticated queries should be performed using specialized search engines such as Entrez at NCBI (<http://www.ncbi.nlm.nih.gov>) or SRS (<http://srs.ebi.ac.uk>).

3. For example, type estradiol receptor into the text box. Figure 5.5.8 shows the result for this query.

The leftmost column shows that there are two PDB entries for estradiol receptor, namely 1qkt and 1qku. The latter has three chains named A, B and C. The second column indicates that the chain 1qkuA is representative of all the chains in the PDB90 set, which retains a style representative for clusters of very similar proteins. The third column shows that 1qkuA belongs to domain fold class 1060. Fold class indices are not stable, i.e., they may change between updates of the Dali database.

4. Click on a link in the Fold column to show a section of the Fold Index. All members of the fold class can be seen here at a glance (Fig. 5.5.9).

Domains in the Fold Index are annotated by the sequence family to which they belong. Sequence families are defined in the ADDA database (Heger and Holm, 2003) based on shared sequence motifs. ADDA unifies many structural neighbors with little overall

Dali fold query: 1060

Fold Index	PDB Code	Adda	Browse	Interact	Compound
1060.1.1.1.1.1	1x1sF_1	313	browse	interact	RETINOIC ACID RECEPTOR RXR-ALPHA
1060.1.1.1.1.1	1xv9B_1	313	browse	interact	RETINOIC ACID RECEPTOR RXR-ALPHA
1060.1.1.1.2.1	1lie9A_1	313	browse	interact	VITAMIN D3 RECEPTOR
1060.1.1.1.3.1	1pq6B_1	313	browse	interact	OXISTEROLS RECEPTOR LXR-BETA
1060.1.1.1.4.1	1uh1B_1	313	browse	interact	RETINOIC ACID RECEPTOR RXR-BETA
1060.1.1.1.5.1	1r1kD_1	313	browse	interact	ECYDOME RECEPTOR
1060.1.1.1.6.1	1mclA_1	313	browse	interact	ORPHAN NUCLEAR RECEPTOR RXR
1060.1.1.1.7.1	1ot7A_1	313	browse	interact	SILE ACID RECEPTOR
1060.1.1.1.8.1	1oshA_1	313	browse	interact	SILE ACID RECEPTOR
1060.1.1.1.9.1	1s0xA_1	313	browse	interact	NUCLEAR RECEPTOR RXR-ALPHA
1060.1.1.1.9.1	1n4hA_1	313	browse	interact	NUCLEAR RECEPTOR RXR-BETA
1060.1.1.1.10.1	1gwoB_1	313	browse	interact	PEROXISOME PROLIFERATOR ACTIVATED REC
1060.1.1.1.11.1	1navA_1	313	browse	interact	HORMONE RECEPTOR ALPHA 1, THR1
1060.1.1.1.11.1	1nq2A_1	313	browse	interact	THYROID HORMONE RECEPTOR BETA-1
1060.1.1.1.12.1	1yjeA_1	313	browse	interact	ORPHAN NUCLEAR RECEPTOR NR4A1
1060.1.1.1.13.1	1pduA_1	313	browse	interact	NUCLEAR HORMONE RECEPTOR NR4A1
1060.1.1.1.14.1	1ov1E_1	313	browse	interact	ORPHAN NUCLEAR RECEPTOR NR1
1060.1.1.1.15.1	1qknA_1	313	browse	interact	ESTROGEN RECEPTOR BETA
1060.1.1.1.16.1	1l2jA_1	313	browse	interact	ESTROGEN RECEPTOR BETA
1060.1.1.1.17.1	1qkuA_1	313	browse	interact	ESTRADIOL RECEPTOR
1060.1.1.1.18.1	1xb7A_1	313	browse	interact	STERIOD HORMONE RECEPTOR ERR1
1060.1.1.1.18.1	1tfcA_1	313	browse	interact	ESTROGEN-RELATED RECEPTOR GAMMA
1060.1.1.1.19.1	1m2zA_1	313	browse	interact	GLUCOCORTICOID RECEPTOR
1060.1.1.1.19.1	1e3kA_1	313	browse	interact	PROGESTERONE RECEPTOR
1060.1.1.1.20.1	1pz1A_1	313	browse	interact	HEPATOCYTE NUCLEAR FACTOR 4-ALPHA
1060.1.1.1.20.2	1lv2A_1	313	browse	interact	HEPATOCYTE NUCLEAR FACTOR 4-GAMMA
1060.1.1.1.21.1	1yucA_1	313	browse	interact	ORPHAN NUCLEAR RECEPTOR NR5A2
1060.1.1.1.21.1	1pk5A_1	313	browse	interact	ORPHAN NUCLEAR RECEPTOR NR5A2
1060.1.1.1.22.1	1s9qB_1	313	browse	interact	ESTROGEN-RELATED RECEPTOR GAMMA
1060.1.1.2.1.1	1g2nA_1	313	browse	interact	ULTRASPIRACLE PROTEIN
1060.1.1.2.1.1	1hg4A_1	313	browse	interact	ULTRASPIRACLE
1060.1.1.2.2.1	21bd_1	313	browse	interact	RETINOIC ACID RECEPTOR GAMMA
1060.1.1.2.2.1	1dkfB_1	313	browse	interact	RETINOID X RECEPTOR-ALPHA
1060.1.1.2.3.1	11bd_1	313	browse	interact	RETINOID X RECEPTOR
1060.1.2.1.1.1	1fa6X_1	313	browse	interact	RETINOIC ACID RECEPTOR RXR-ALPHA
1060.1.2.1.2.1	1kkqA_1	313	browse	interact	PEROXISOME PROLIFERATOR ACTIVATED REC
1060.1.2.1.3.1	1xdkF_1	313	browse	interact	RETINOIC ACID RECEPTOR RXR-ALPHA
1060.1.2.1.4.1	1av9A_1	313	browse	interact	RXR RETINOID X RECEPTOR
1060.1.2.1.5.1	1xj7A_1	313	browse	interact	ANDROGEN RECEPTOR
1060.1.2.1.6.1	1hj1A_1	313	browse	interact	ESTROGEN RECEPTOR BETA
1060.1.2.2.1.1	1pq6C_1	313	browse	interact	OXISTEROLS RECEPTOR LXR-BETA

Figure 5.5.9 A large number of nuclear receptors belonging to the same fold class as estradiol receptor. Where a sequence-structure-domain mapping is available, they have all been classified into the same ADDA domain family (numbered 1060).

sequence similarity in terms of percent identity. As can be seen from Figure 5.5.9, the nuclear receptors are unified by ADDA into one family.

ADDA family indices are not stable; that is, they may change between releases of the ADDA database.

- Go back to the previous page (Fig. 5.5.8) and click on the “interact” link to see details about the structural neighbors of each domain. The list of neighbors of estradiol receptor is shown in Figure 5.5.10.

The hits are ranked by Z-Score with best hits at the top of the table. As a general rule, a Z-score above 20 means the two structures are definitely homologous, between 8 and 20 means the two are probably homologous, between 2 and 8 is a grey area, and below 2 is not significant. When structural similarity is due to homology, the proteins often have related biochemical functions, e.g., in Figure 5.5.10 the top hits are all nuclear receptors.

Other listed parameters in Figure 5.5.10 are as follows: %ide (percentage amino acid identity in aligned positions); rmsd (root-mean-square deviation of C α atoms in superimposition); lali (number of structurally equivalent positions); and lseq2 (length of the structural neighbor).

- To display structural alignments between estradiol receptor and its neighbors as one-dimensional alignments or in three-dimensional superimposition, select a few structures by clicking on check boxes on the left. Then click on the Structure Alignment button, which results in a multiple structure alignment page (Fig. 5.5.11) similar to a sequence alignment. Secondary structure definitions are shown below the amino acid sequences.

neighbour	Z	%ide	rmsd	lali	lseq2	PDB	compound
0: 1qkuA	44.6	100	0.0	250	250	PDB	ESTRADIOL RECEPTOR
1: 1qknA	30.4	57	1.3	219	228	PDB	ESTROGEN RECEPTOR BETA
2: 1tfcA	30.1	36	1.8	225	226	PDB	ESTROGEN-RELATED RECEPTOR GAMMA
3: 1l2lA	28.3	54	1.9	224	232	PDB	ESTROGEN RECEPTOR BETA
4: 1xb7A	27.8	33	1.7	214	215	PDB	STEROID HORMONE RECEPTOR ERR1
5: 1a9aB	27.1	35	1.7	205	216	PDB	ESTROGEN-RELATED RECEPTOR GAMMA
6: 1e3kA	26.7	22	1.9	229	251	PDB	PROGESTERONE RECEPTOR
7: 1m2aA	26.4	25	2.0	228	255	PDB	GLUCOCORTICOID RECEPTOR
8: 1pk5A	24.4	9	5.9	95	242	PDB	ORPHAN NUCLEAR RECEPTOR NR5A2
9: 1p2lA	23.8	25	0.0	212	227	PDB	HEPATOCTE NUCLEAR FACTOR 4-ALPHA
10: 1c1kD	23.7	18	0.0	221	236	PDB	ECDYSONE RECEPTOR
11: 1p46B	23.3	18	2.3	222	241	PDB	OXYSTEROLS RECEPTOR LXR-BETA
12: 1lv2A	23.2	26	2.2	207	225	PDB	HEPATOCTE NUCLEAR FACTOR 4-GAMMA
13: 1vuca	23.2	25	2.9	219	240	PDB	ORPHAN NUCLEAR RECEPTOR NR5A2
14: 1aduA	23.0	21	2.7	219	230	PDB	NUCLEAR HORMONE RECEPTOR HR38
15: 2lba	22.7	22	2.6	217	238	PDB	RETINOIC ACID RECEPTOR GAMMA
16: 1ie9A	22.3	21	0.0	223	255	PDB	VITAMIN D3 RECEPTOR
17: 1nd4A	22.2	18	2.4	217	244	PDB	NUCLEAR RECEPTOR ROR-BETA
18: 1uh1B	22.2	17	2.5	212	219	PDB	RETINOIC ACID RECEPTOR RXR-BETA
19: 1ov1B	22.0	23	2.7	215	250	PDB	ORPHAN NUCLEAR RECEPTOR NURR1 (MSE 414, 496)
20: 1navA	21.9	16	2.5	215	253	PDB	HORMONE RECEPTOR ALPHA 1, THR1
21: 1nq2A	21.9	17	0.0	213	255	PDB	THYROID HORMONE RECEPTOR BETA-1
22: 1a0aA	21.9	17	2.4	214	251	PDB	NUCLEAR RECEPTOR ROR-ALPHA
23: 1hq4A	21.7	26	2.5	203	240	PDB	ULTRASPIRACLE
24: 1v1aA	21.6	25	2.9	208	226	PDB	ORPHAN NUCLEAR RECEPTOR NR4A1
25: 1x1aF	21.5	19	2.8	221	242	PDB	RETINOIC ACID RECEPTOR RXR-ALPHA
26: 1a2aA	21.3	26	3.0	204	246	PDB	ULTRASPIRACLE PROTEIN
27: 1oshA	21.1	18	2.3	202	216	PDB	BILE ACID RECEPTOR
28: 1a57A	21.0	19	2.7	213	229	PDB	BILE ACID RECEPTOR
29: 1dk7B	21.0	19	2.7	208	232	PDB	RETINOID X RECEPTOR-ALPHA
30: 1xv9B	20.5	20	2.7	220	246	PDB	RETINOIC ACID RECEPTOR RXR-ALPHA
31: 1xw9B	20.4	15	2.9	222	271	PDB	PEROXISOME PROLIFERATOR ACTIVATED RECEPTOR (PPAR)
32: 1nrc1A	20.3	15	2.6	213	278	PDB	ORPHAN NUCLEAR RECEPTOR RXR
33: 1lba	19.0	29	2.9	194	238	PDB	RETINOID X RECEPTOR
34: 1kkaA	18.1	15	2.9	214	269	PDB	PEROXISOME PROLIFERATOR ACTIVATED RECEPTOR

Figure 5.5.10 Clicking on the “interact” link in Figure 5.5.8 or 5.5.9 leads to the list of structural neighbors of estradiol receptor. Hits 1-34 are members of the same fold class comprising nuclear receptors. Hits further down the list have a much lower Z-score than the nuclear receptors and represent biologically noninteresting hits that match in a helical bundle motif.

Typically secondary structure assignments agree very well even though sequence identity may be low (see Fig. 5.5.10).

The Structure/Sequence Alignment button, shown in Figure 5.5.10, augments the structural alignment by additionally displaying related sequences, which are detected by PSI-Blast and stored in the ADDA database (Heger and Holm, 2003). This view is useful for checking sequence patterns that are conserved across distantly related protein families. Conserved functional sites are a strong hint at common evolutionary origins.

In the alignment, residues are colored if the frequency of the amino acid type in the column is above 50%.

- Go back to the previous page and click on the 3D Superimposition button to view the superimposed C α traces of the selected structures in 3D using RasMol or another PDB viewer. The 3D Superimposition button launches a RasMol script if the browser is appropriately configured. Use the “PDB format” button to download the C α coordinates of selected neighbors superimposed onto the query structure.

Make external links to the Dali database

- External sites may be linked directly to the query engine of the Dali database. To make a link from a PDB identifier to the database, use the call http://www.bioinfo.biocenter.helsinki.fi/daliquery?search_term, where the search_term is a PDB identifier (e.g., 2kau or 2kauC).

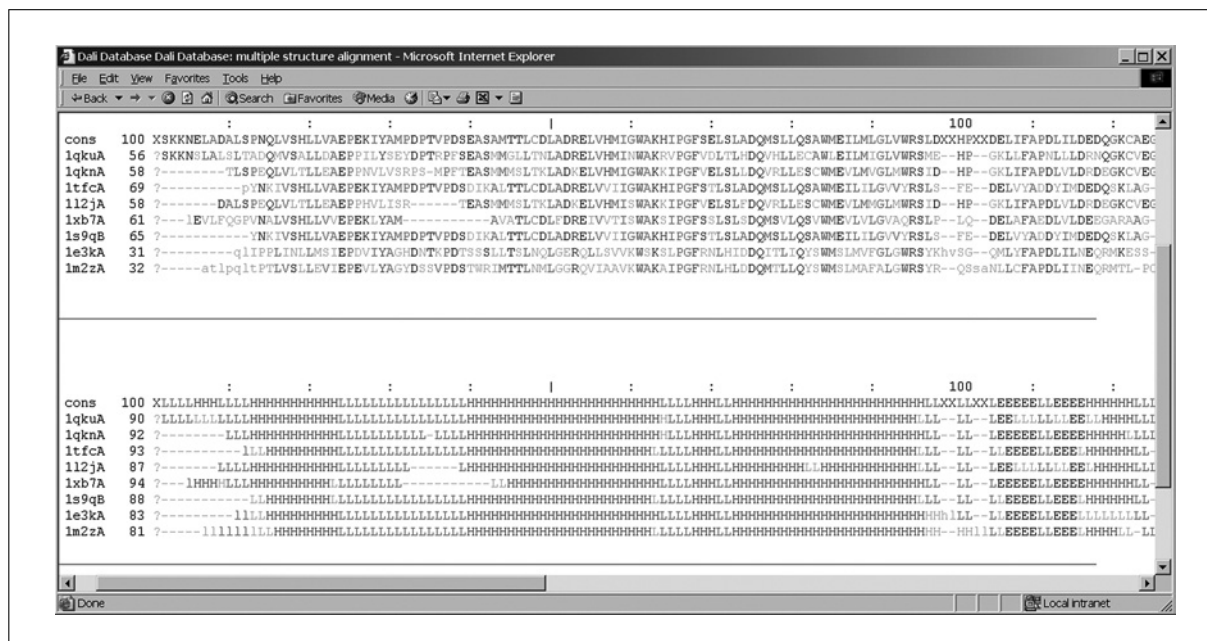


Figure 5.5.11 Multiple structure-alignment of estradiol receptor and selected structural neighbors. Notation: three-state secondary structure definitions by DSSP (reduced to H=helix, E=sheet, L=coil) are shown below the amino acid sequences. For the color version of this figure go to <http://www.currentprotocols.com>.

Download data from the Dali database

- For noninteractive use, comprehensive computer-readable database-dumps are provided for large-scale studies. These are accessed from the link to Downloads from the home page of the Dali database (<http://www.bioinfo.biocenter.helsinki.fi/dali/start>).

ALTERNATE PROTOCOL 1

COMPARING TWO STRUCTURES USING THE STAND-ALONE VERSION OF DaliLite

This simple protocol is the command-line version of that performed online by the DaliLite server for pairwise structure comparison (Basic Protocol 1). The inputs are two protein structures in PDB format. The output is a set of HTML files, which should be viewed from a browser. Rough timings are from a few seconds up to tens of seconds per pairwise comparison.

Necessary Resources

Hardware

Computer that operates the Linux operating system (e.g., Sun, Alpha, Silicon Graphics, PC)

Software

DaliLite program (see Support Protocol)

Perl interpreter (Perl v. 5.0 or higher; <http://www.perl.org>)

Internet browser (e.g., Internet Explorer, <http://www.microsoft.com>; Netscape, <http://browser.netscape.com>; or Firefox, <http://www.mozilla.org/firefox>)

Files

Two protein structures in PDB format files

- Download and install DaliLite as described in the Support Protocol.

Using Dali for
Structural
Comparison of
Proteins

5.5.12

2. The option to run DaliLite is DaliLite -pairwise <pdbfile1> <pdbfile2>, where the arguments <pdbfile1> <pdbfile2> should be replaced by the PDB file names entered as user input after the Linux prompt as in the example below.

```
Linux-prompt> perl DaliLite -pairwise /pdb/1wsy.brk  
/pdb/2kau.brk > log
```

```
Linux-prompt> netscape index.html
```

3. The program computes the structural alignments for all chains in pdbfile1 against all chains in pdbfile2 and creates a set of HTML pages linked from the top page "index.html". The first structure is called "mol1" and the second, "mol2". All data are stored in the current work directory, overwriting any previous results generated using this option. The output is identical to that from Basic Protocol 1 (Figs. 5.5.2 through 5.5.4).

COMPARING LARGE SETS OF STRUCTURES USING THE STAND-ALONE VERSION OF DaliLite

ALTERNATE PROTOCOL 2

This is a more advanced protocol that allows the systematic comparison of large sets of structures using the stand-alone version of DaliLite. It performs the structural comparisons between all pairs of two user-provided lists of structures. The results are stored in an internal alignment format which can be processed by computer programs for further statistical analysis. There is an option to reformat the results as "human-readable" output.

Necessary Resources

Hardware

Computer that operates the Linux operating system (Sun, Alpha, Silicon Graphics, PC)

Software

DaliLite program (see Support Protocol)

Perl interpreter (Perl v. 5.0 or higher, <http://www.perl.org>)

Internet browser (e.g., Internet Explorer, <http://www.microsoft.com>; Netscape, <http://browser.netscape.com>; or Firefox, <http://www.mozilla.org/firefox>)

Files

Protein structures in PDB format

1. Download and install DaliLite as described in the Support Protocol.

Prepare structures

2. Prepare all structures that one wants to compare using the -readbrk option, supplying a unique identifier for the structure as the second argument as follows.

```
Linux-prompt> perl DaliLite -readbrk <pdbfile> <pdbid>
```

The identifier must be in PDB style, i.e., four characters long, as shown in the examples below.

```
DaliLite -readbrk 3ubp.brk 3ubp
```

```
DaliLite -readbrk /data/pdb/3ubp.brk 3ubp
```

```
DaliLite -readbrk /data/pdb/pdb3ubp.ent 3ubp
```

These structural data are stored in a DAT subdirectory under the DaliLite home directory.

Modeling Structure from Sequence

5.5.13

3. The program automatically generates a data file for each chain in the PDB entry. In the above examples, 3ubpA.dat, 3ubpB.dat, and 3ubpC.dat are created in the DAT subdirectory. The system uses the DSSP program by Kabsch and Sander (included in the DaliLite distribution package) to parse the information out of the PDB file.

DSSP requires that the complete backbone (N, C α , C, O atoms) is present or it will skip the residue. The MaxSprout server (<http://www.ebi.ac.uk/maxsprout>) can be used to build full coordinates from a C α trace.

4. The DAT file includes information about the C α coordinates, primary structure, secondary structure elements (from DSSP, Kabsch and Sander, 1983), and putative folding pathway of the protein (from PUU, Holm and Sander, 1994). The first line of a properly formed DAT file is shown in Figure 5.5.12.

If reading of the coordinates fails for any reason, only zeros will appear on the first line of the DAT file.

Generate structural alignments

5. There are options for pairwise, one-against-many, and many-against-many comparisons. The structures are specified using the unique identifiers, introduced in step 2 when reading in PDB structures using the `–readbrk` option.

Pairwise alignments of two structures are generated using exhaustive search (Parsi method). If the query structure has few secondary structure elements, the program automatically switches to the Soap method. Monte Carlo optimization is used for refinement (see Table 5.5.2).

6. DaliLite has three main options for alignment. The simplest is pairwise alignment (`–align` option) which takes two chain identifiers as argument, for example:

```
Linux-prompt> perl DaliLite –align 3ubpC 1gkpA
```

The arguments are the unique identifier with the chain identifier appended. Alignment data is automatically output to alignment files: `<code>.dccp`.

7. An optimal and a number of suboptimal structural alignments are reported for each pair of structures. Similarities with a Z-score below zero are omitted from the output. The format is shown and explained in Figure 5.5.13.

>>>>	1xg8A	108	7	3	4	EHEHEEH
						order of secondary structure elements
						number of beta-strands (E)
						number of helices (H)
						total number of secondary structure elements
						number of residues
						chain identifier

Figure 5.5.12 Format of the DAT file.

Table 5.5.2 Program Modules of the Dali Suite

Program	Purpose	Reference
DSSP	Parse PDB entry; define secondary structure elements	Kabsch and Sander (1983)
PUU	Derive a tree of compact substructures to guide alignment	Holm and Sander (1994)
Wolf	Very fast filter to identify obvious similarities	Holm and Sander (1995)
Soap	Align structures with little secondary structure	Falicov and Cohen (1996)
Parsi	Sensitive branch-and-bound alignment algorithm	Holm and Sander (1996)
Dalicon	Refine all alignments generated by the above methods (with different objective functions) using a Monte Carlo algorithm that maximizes the Dali score	Holm and Sander (1993)

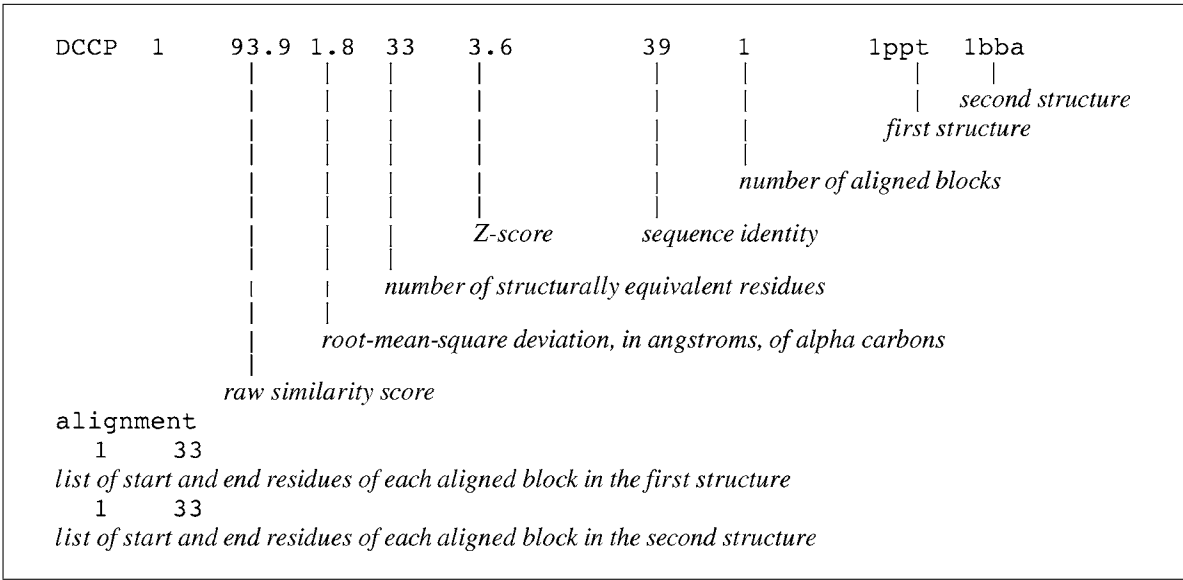


Figure 5.5.13 Format of the DCCP file.

8. Prepare a list of chain identifiers in a file to perform a pairwise comparison of the query to each structure in the list. For example, the list file “mylist” may have the following contents.

```
1bf6A
1j79A
1a4mA
1k70A
3ubpC
```

9. To compare 3ubpC against each entry in the list file, enter the following user input after the Linux prompt.

```
Linux-prompt> perl DaliLite -list 3ubpC mylist
```

10. For all-against-all comparison enter the following user input after the Linux prompt.

```
Linux-prompt> perl DaliLite -AllAll mylist
```

The database search option (-search) uses the same shortcuts as the Dali server. Note that using this option is dependent on an up-to-date list of representative structures and the complete database of precomputed structural alignments. This database resides in the DCCP/ subdirectory. Updates of the database are available for download. Click the Downloads link on the home page of the Dali database <http://www.bioinfo.biocenter.helsinki.fi/dali/start>.

11. Convert the alignment file (files with the extension .dccp in DaliLite's internal format) to a readable format using the -format option.). The arguments to the -format option are the identifier of the query structure, the alignment datafile, a listfile of valid identifiers, and the name of the output file illustrated in the following command.

```
Linux-prompt> perl DaliLite -format 3ubpC 3ubpC.dccp
representatives.list 3ubpC.html
```

Only comparisons to structures listed in the listfile will be output.

12. The output file is in HTML format. It contains the list of structural neighbors and links to the structural alignments similar to Figure 5.5.2.
13. To construct a similarity matrix of a large set of proteins, extract the DCCP lines from the alignment data files (* .dccp).

The similarity matrix can be used as input data for hierarchical clustering.

Note that several alternative alignments may be reported by protein pair.

SUPPORT PROTOCOL

DOWNLOADING AND INSTALLING THE DaliLite STAND-ALONE PROGRAM

DaliLite is a stand-alone program package that can help researchers compare large numbers of protein structures for specialized projects efficiently and locally. The DaliLite distribution package contains a self-contained package of scripts and programs written in Perl and Fortran 77. It has been tested on the Linux operating systems (RedHat distribution, version 6.0; <http://www.redhat.com>) and on Cygwin, a Linux-like environment for Microsoft Windows (<http://cygwin.com>). The program code is distributed to academic users. Commercial use is prohibited.

Necessary Resources

Hardware

Computer that operates the Linux operating system (e.g., Sun, Alpha, Silicon Graphics, PC)

Software

Fortran 77 compiler (<http://www.gnu.org/software/fortran/fortran.html>)

Perl interpreter (Perl v. 5.0 or higher <http://www.perl.org>)

Cygwin (<http://cygwin.com>), optional

1. Download the academic license agreement from http://www.bioinfo.biocenter.helsinki.fi/dali_lite/downloads, and print, sign, and fax it to the address indicated.
2. Download the DaliLite program package by clicking on the link at the top of the above Web page.

The current distribution version (as of this writing) is 2.4.2.

Complete instructions for compilation and installation are available in the INSTALL file included in the DaliLite distribution, as well as instructions for where to obtain the necessary software resources. Test examples are included in the distribution package.

- 3a. *To unpack the distribution package using Linux:* Enter the following user input after the Linux prompt.

```
Linux-prompt> tar -zxvf DaliLite_2.4.2.tar.gz
```

```
Linux-prompt> cd ./DaliLite_2.4.2/Bin
```

- 3b. *To unpack the distribution package using Cygwin:* Enter the following user input after the Linux prompt.

```
Linux-prompt> mv -f Makefile_cygwin Makefile
```

4. Use a text editor to set proper HOMEDIR and ESCAPED_HOMEDIR in Makefile by typing the following commands.

```
Linux-prompt> make clean
```

```
Linux-prompt> make install
```

```
Linux-prompt> make test
```

```
Linux-prompt> cd ../
```

```
Linux-prompt> ./DaliLite -help
```

Note that the maximum acceptable length of the HOMEDIR path is 70 characters.

GUIDELINES FOR UNDERSTANDING RESULTS

As in sequence analysis, the goal of structural database searching is usually to identify homologous proteins that might provide clues to the function of the query protein. Homology means descent from a common ancestor. One can infer homology from sequence or structural similarities that are so strong they would not be expected to have arisen by chance. The structural neighbors reported by Dali (Basic Protocol 2) are ranked in order of decreasing structural similarity (Z-score). Basic Protocol 3 allows browsing a precomputed clustering of all structures into groups with similar folds. The clustering is hierarchical, so that the most similar structures are found near the tips of the “fold tree,” and more general similarities of fold types are found nearer the root. The organization of fold space is based on Z-scores.

The Z-Score is the most important measure of quality of the structural alignment. Homologous proteins cluster at the top of the ranked list, but the boundary between homologous and unrelated proteins varies from one family to another. As a general rule, a Z-score above 20 means the two structures are definitely homologous, between 8 and 20 means the two are probably homologous, between 2 and 8 is a grey area, and a Z-Score below 2 is not significant.

The size of the proteins influences Z-scores: small structures will tend to have small Z-Scores, whereas a medium Z-Score for very large structures need not imply a biologically interesting relationship. Fold type also has an effect: α/β proteins also usually have higher Z-scores than all- β proteins. For example, TIM barrel proteins have about sixteen secondary structure elements in a similar ($\beta\alpha$) 8-barrel topology and are unified at Z-scores above 10. In contrast, two small avian polypeptides (PDB codes 1ppt and 1bba) contain only one helix and a proline-rich loop and get a Z-score around 4. In view of the Z-score, it is much more improbable to observe sixteen helices and strands arranged in a similar fold than to find a similar arrangement of just a helix and a loop.

Homologous proteins often share significant functional similarities. An attempt should be made to place the query structure in the context of a fold similarity dendrogram as in Figure 5.5.6 before transferring function. There is always a best hit. Reciprocal nearest neighbors suggest more similar functions than if the query protein joins a whole branch of functionally diverse proteins. For example, in the receptor dendrogram (Fig. 5.5.6), sex hormone receptors form one subcluster while the orphan receptor is about equidistant from all the other receptors.

RMSD is a measure of the average deviation in distance between aligned alpha carbons. For sequences sharing 50% identity, this should be around 1.0. Dali maximizes a geometrical similarity score, which is defined in terms of similarities of intramolecular distances and is thus not primarily aiming to generate alignments with low RMSD. The RMSD and number of equivalent residues (NE) are reported because they are traditional measures. Note that an alignment is considered better if it has both a smaller RMSD and a larger NE. If both RMSD and NE are smaller or both are larger, it is not possible to establish an order between the alignments.

It is generally assumed that if two sequences share over 40% identity, then they are unambiguously homologous. However, two distantly related proteins may share very low sequence identity but still be homologous, and conversely, two sequences may locally share as much as 30% identity but be unrelated. Therefore, the percentage of sequence identity is only a guide.

In lieu of numbers, it is often informative to inspect using RasMol or another graphics program, whether the structurally equivalent regions form a continuous, compact structural core. If there are many known structures in a superfamily, secondary structure elements will line up consistently in the multiple structure alignment views (Fig. 5.5.11). Check especially for the conservation of known active site residues. Conservation profiles can be studied in multiple sequence alignments of protein families in sequence classification databases such as the Automatic Data Decomposition Algorithm (ADDA) at <http://www.bioinfo.biocenter.helsinki.fi/sqgraph/pairsdb> or PFAM (<http://www.sanger.ac.uk/Pfam>). Enzyme superfamilies have sharp signatures but binding domains can have very little sequence similarity. Without a sequence signature, it is harder to establish homology.

COMMENTARY

Background Information

The rapidly growing number of known tertiary structures makes protein structure comparison important. In the center of biological interest are evolutionary relationships inferred from quantifiable similarities between proteins. Sequence similarity searches are able to detect evolutionary relationships down to a sequence identity of about 25%. Below this level of sequence identity starts the “twilight zone” of similarity. Comparing structures can help to extend the validity of an evolutionary relationship between proteins through this zone. This is because the structure of proteins is much better preserved during evolution than the sequence (Chothia and Lesk, 1986). By searching structural databases, molecular biologists can gain a considerable amount of information about connections between pro-

tein families that are unseen using sequence alone. The prediction of protein function based on structure aims at the unification of protein families into larger sets (superfamilies). Functionally divergent families classified into the same superfamily typically exploit a conserved mechanical or biochemical mechanism that has been adapted to different cellular processes and substrates (Holm and Sander, 1996). Inferring complex conserved properties is the basic reason for providing the systematic structure-structure comparison and classification of available proteins.

Improved methods of protein engineering, crystallography and NMR spectroscopy have led to a surge of new protein structures deposited in the Protein Data Bank (PDB). At the end of 2004, the PDB contained over 28,000 protein structures, and the structural genomics

initiative aims to provide a structure for each major protein family within a decade. This wealth of data needs to be organized and correlated using automated methods. Nearly all proteins have structural similarities to other proteins. General similarities arise from principles of physics and chemistry that limit the number of ways in which a polypeptide chain can fold into a compact globule. Evolutionary relationships result in surprising similarities (which are even stronger than similarity due to convergence caused by physical principles). Because structure tends to diverge more conservatively than sequence during evolution, structure alignment is a more powerful method than pairwise sequence alignment for detecting homology and aligning the sequences of distantly related proteins. In favorable cases, comparing 3D structures may reveal biologically interesting similarities that are not detectable by comparing sequences and may help to infer functional properties of hypothetical proteins.

Automatic methods enable exhaustive all-against-all structure comparisons. As a result, each structure in the PDB can be represented as a node in a graph where similar structures are neighbors of each other and structurally unrelated proteins are not neighbors. Clustering the graph at different levels of granularity removes redundancy and aids navigation in protein space. At long range, the overall distribution of folds is dominated by secondary structure composition (e.g., all alpha or alternating alpha/beta). At intermediate range, clusters are related by shape similarity that does not necessarily reflect similarity of biological function (for example, globins and colicin A). At close range, clusters represent protein families related through strong functional constraints (for example, hemoglobin and myoglobin). Evolutionary relationships can be recovered by searching for continuous neighborhoods (Dietmann and Holm, 2001).

In order to identify natural groupings of any set of objects, one needs a measure of distance or similarity. Structure comparison programs derive a structural alignment, which maximizes similarity or minimizes distance. The alignment defines a one-to-one correspondence of amino acid residues (sequence positions) in two proteins. This is analogous to sequence alignment except that the notion of similarity or dissimilarity is much more complex between three-dimensional objects than between linear strings. For example, the conformation of a point mutant usually differs from the wild-type protein only locally and

only by a few tenths of an angstrom. Much larger deviations are commonly observed in pairs of homologous proteins, and with increasing sequence dissimilarity small shifts in the relative orientations of secondary structure elements accumulate and reach several angstroms and tens of degrees. At the largest evolutionary distances, only the topology of the fold or folding motif is conserved. (Topology here means the relative location of helices and strands and the loop connections between them.) Deviations can be even larger and qualitatively different when structural similarity is the result of convergent rather than divergent evolution. In particular, convergent evolution may result in similar 3D folds that differ in the topology of loop connections. The modular architecture of proteins presents another complication. Large proteins can be decomposed into semiautonomous, globular folding units called domains. Domains are often evolutionarily mobile modules and may carry specific biological functions. Because a common domain may be surrounded by completely unrelated domains, most structure comparison methods search for local similarities.

Given a measure of similarity or distance, the algorithmic problem is to find the set of corresponding points in two structures that optimize this target function. Just as there is much latitude in the formulation of the structure comparison problem, many different types of optimization algorithms have been employed. Similarity measures of the sum-of-pairs form and subgraph isomorphism formulations of the structure comparison problem belong to the NP-complete class of problems and one has to resort to heuristics for practical algorithms. Heuristic approaches do not aim for provably correct solutions, gaining computational performance at the potential cost of accuracy or precision. Many programs use a hierarchical approach, where promising seeds for alignment are identified using local criteria based on dynamic programming, distance difference matrices, maximal common subgraph detection, fragment matching, geometric hashing, unit vector comparison, or local geometry matching (reviewed by Sierk and Kleywegt, 2004). The initial set of correspondences is then optimized globally using methods such as double dynamic programming, Monte Carlo algorithms or simulated annealing, a genetic algorithm, or combinatorial searching. Recently, it has been proved that brute force, exhaustive scanning of the six degrees of freedom from rotations and translations in rigid-body superimposition leads to

a polynomial-time approximation algorithm for the problem of determining the maximum number of C α atom pairs that can be superimposed within a given RMSD at a given error. However, this solution is too computationally demanding for practical application (Kolodny and Linial, 2004).

The Dali method is based on a sensitive measure of geometrical similarities defined as a weighted sum of similarities of *intramolecular* distances (see the appendix for details). Three-dimensional shape is described with a matrix of all intramolecular distances between the C α atoms. Such a distance matrix is independent of coordinate frame but contains more than enough information to reconstruct the 3D coordinates, except for overall chirality, by distance geometry methods. Imagine sliding a (transparent) distance matrix on top of another one. Depending on the register of the two matrices, similar substructures will stand out as submatrices with similar patterns. Structurally equivalent regions can be filtered out with a fixed cutoff on acceptable differences of intramolecular distances or, as the authors prefer, with a continuous function defined in terms of relative distance deviations. The common structure is revealed when two distance matrices are brought into register by keeping only rows or columns corresponding to the structurally equivalent residues (Fig. 5.5.14).

The Dali program has a modular architecture, where the structure alignment/database searching problem is approached by a cascade of algorithms. The Dali package consists of many Fortran programs and Perl5 scripts. The program flow is controlled by a Perl wrap-

per script that calls other programs as needed. Each program implements pairwise structure comparisons using different algorithms. References for these programs are given in Table 5.5.2. The goal of a database search is to find all structures that are significantly similar to the query. A conceptual map of fold space is determined by the precomputed all-against-all structural alignments between all representative structures. Based on this map, the database search by the Dali server tries shortcuts to quickly place the query structure in a “known” location of fold space. If a strong match is found to one database structure, then the search can be restricted to the precomputed neighborhood of this structure. Fast but approximate methods can quickly find obvious structural resemblances. Slower but most sensitive algorithms need then only be applied to a smaller set of candidates. DaliLite has the core algorithmic functionality of the Dali server. The DaliLite programs perform systematic pairwise comparisons without shortcuts and can therefore be run independently of database updates.

Applications

The exponential growth in the number of newly solved protein structures makes correlating and classifying the data an important task. Dali is now used routinely by crystallographers worldwide to screen the database of known structures for similarity to newly determined structures. The application of Dali to newly released structures led to a string of discoveries of unexpected distant evolutionary relationships. For example, a remarkably diverse set of distant relatives of urease were

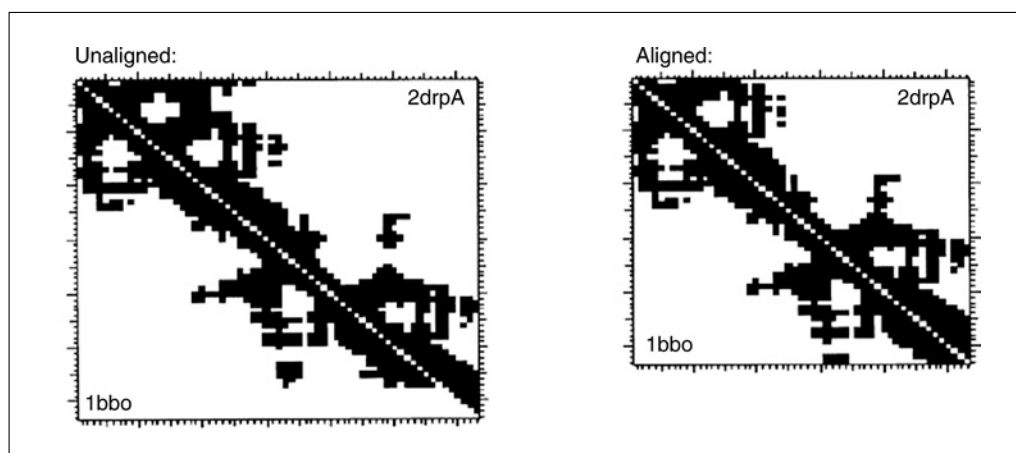


Figure 5.5.14 Distance matrix representations. Unaligned: Distance matrix representation of two different proteins, one in the upper and the other in the lower triangle. Aligned: Structural alignment identifies a one-to-one correspondence between a subset of residues. The respective submatrices of the distance matrix display similar contact patterns.

identified based on structural and sequence analysis (Holm and Sander, 1997); several blind fold predictions have since been verified by experimental structure determination.

Comparison to other techniques

Dali was ranked at the top among seven protein structure comparison methods and two sequence comparison programs that were evaluated on their ability to detect either protein homologues or domains with the same topology (fold) as defined by the CATH structure database (Novotny et al., 2004).

Critical Parameters

The Dali program has been run successfully with default parameters since its inception (Holm and Sander, 1993). The results usually agree quite well with human experts' assessments. For example, the dendrogram of structural similarities by Dali has similar topology to the SCOP hierarchical classification based on visual analysis and biological knowledge (Dietmann and Holm, 2001).

While the authors strongly advise against changing parameter values from their default values, a description of the numerical parameters that go into the algorithms is given in the appendix.

Troubleshooting

Similarity not reported

The Dali system reports only similarities above an empirically chosen threshold of $Z = 2$. This captures most cases of topological similarity of globular domains. However, in some fold types structural similarities between parts of globular domains also score above this threshold.

Known similarity not reported

The Dali server currently reports similarities only to PDB25 representatives. The purpose of using PDB25 is to suppress the redundancy of output due to multiple structure determinations of mutants or of the same protein in slightly differing conditions. Thus, a particular PDB entry, known to be structurally similar to the query, might appear to be missing from the output list only because the representative structure is a different PDB entry. The Dali database reports similarities between PDB90 representatives. The PDB90 representatives for any PDB entry can be found by using the search functionality on the homepage of the Dali database (<http://www.bioinfo.biocenter.helsinki.fi/dali>).

Empty result

The Dali database includes all peptide chains from the PDB, except C α -only entries and chains that are shorter than thirty residues. DaliLite requires that the backbone atoms (N, C α , C, O) must be complete. The user can build a complete backbone model from the C α trace using the MaxSprout Server. The Dali server runs MaxSprout automatically, if only a C α trace is submitted. The submission to the Dali server will fail unless the message is plain text, as encoded messages (e.g., MIME or BinHex) are rejected by the server.

Complex comparison

Each chain is compared separately. For example, similarities to structural units made up of a dimer of two different chains (e.g., A and B) will not be detected. There is a way around this limitation, which requires manual editing of the PDB entry by the user: renumber the residues in a sequential order and give all chains the same chain identifier.

Multidomain proteins

It is advisable to break a multidomain query structure into its constituent domains, because the Dali server is designed to report all matches only to the first-found structural neighborhood. That is, if the query protein has one common domain that is found by the fast filters, the search termination criteria are satisfied without a more unique domain in the same query being tested systematically.

Which Z-score threshold implies homology?

This varies for each protein family (Dietmann and Holm, 2001). The topology of the fold dendrogram (hierarchical clustering of domains based on structure similarity) represents evolutionary relationships fairly faithfully, so that homologous structures are found collected in one branch of the tree. However, the borders of the homologous families might be found at Z-scores around 4 (helix-turn-helix DNA-binding domains) or around 14 (TIM barrels).

Technical failures

The Dali server at the EBI is running automatically with minimal human administrative effort. The assumption that the fold space graph is complete is critical to exhaustive database searching but can sometimes be violated for the following reasons: unpredictable failure of the database update (blackouts, computer crashes, network failures, over-running disk space, etc.), failure to process the PDB entry (for example, chains longer than 1000

residues are not handled well), or program bugs. Please report unexpected behavior to dali-help@ebi.ac.uk.

Literature Cited

- Chothia, C. and Lesk, A.M. 1986. The relation between the divergence of sequence and structure in proteins. *EMBO J.* 5:823-826.
- Dietmann, S. and Holm, L. 2001. Identification of homology in protein structure classification. *Nat. Struct. Biol.* 8:953-957.
- Falicov, A. and Cohen, F.E. 1996. A surface of minimum area metric for the structural comparison of proteins. *J. Mol. Biol.* 258:871-892.
- Heger, A. and Holm, L. 2003. Exhaustive enumeration of protein domain families. *J. Mol. Biol.* 328:749-767.
- Holm, L. and Sander, C. 1993. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.* 233:123-138.
- Holm, L. and Sander, C. 1994. Parser for protein folding units. *Proteins* 19:256-268.
- Holm, L. and Sander, C. 1995. 3-D lookup: Fast protein structure database searches at 90% reliability, pp. 179-187. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, Calif.
- Holm, L. and Sander, C. 1996. Mapping the protein universe. *Science* 273:595-602.
- Holm, L. and Sander, C. 1997. An evolutionary treasure: Unification of a broad set of amidohydrolases related to urease. *Proteins* 28:72-82.
- Kabsch, W. and Sander, C. 1983. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22:2577-2637.
- Kolodny, R. and Linial, N. 2004. Approximate protein structural alignment in polynomial time. *Proc. Natl. Acad. Sci. U.S.A.* 101:12201-12206.
- Novotny, M., Madsen, D., and Kleywegt, G.J. 2004. Evaluation of protein fold comparison servers. *Proteins* 54:260-270.
- Sierk, M.L. and Kleywegt, G.J. 2004. Deja vu all over again: Finding and analyzing protein structure similarities. *Structure* 12:2103-2111.

Key References

- Holm and Sander. 1993. See above.
The original Dali reference.
- Holm and Sander, 1996. See above.
Reviews structure comparison methodology, key results, and implications.
- Holm, L. and Park, J. 2000. DaliLite workbench for protein structure comparison. *Bioinformatics* 16:566-567.
The main DaliLite reference, which should be cited in any publication using DaliLite results.

Internet Resources

- <http://www.ebi.ac.uk/DaliLite>
The interactive DaliLite server for comparing two structures to each other and visualizing the structural superimposition.
<http://www.ebi.ac.uk/dali>
The Dali e-mail server for comparing a new structure against the database of known structures.
<http://www.bioinfo.biocenter.helsinki.fi/dali>
The Dali database for browsing structural and sequence neighbors of proteins.
<http://www.bioinfo.biocenter.helsinki.fi/sqgraph/pairsdb>
The ADDA classification assigns every residue of known protein sequences into a domain family and interactively visualizes the sequence neighbors of any query protein in a multiple alignment.
<http://srs.ebi.ac.uk>
<http://www.ncbi.nlm.nih.gov>
SRS at EBI and Entrez at NCBI are comprehensive search engines that cross-reference the PDB identifier of a protein to many other databases.

Contributed by Liisa Holm, Sakari

Kääriäinen, and Chris Wilton

Institute of Biotechnology

University of Helsinki

Helsinki, Finland

Dariusz Plewczynski

Interdisciplinary Centre for Mathematical

and Computation Modeling

University of Warsaw

Warsaw, Poland

APPENDIX

Objective Function

The objective function of the Dali algorithm and the normalization of structural similarity scores to obtain the Z-score are described below.

Consider two proteins labeled A and B. The match of two substructures is evaluated using an additive similarity score S of the form:

$$S = \sum_{i=1}^L \sum_{j=1}^L \varphi(i, j)$$

Equation 5.5.1

where i and j label residues, L is the number of matched pairs (the size of each substructure), and φ is a similarity measure based on some pairwise relationship, in this case, on the C α -C α distances d_{ij}^A, d_{ij}^B . Unmatched residues do not contribute to the overall score. For a given functional form of $\varphi(i, j)$, the largest value of S corresponds to the optimal set of residue equivalences.

Structural similarity algorithms, in this case, search for the largest common substructure between two proteins, but one needs to define a similarity measure that balances two contradictory requirements: maximizing the number of equivalenced residues and minimizing structural deviations. The use of relative rather than absolute deviations of equivalent distances is tolerant to the cumulative effect of gradual geometrical distortions. In Dali, the residue-pair score φ has the form of the equation:

$$\varphi(i, j) = \left(\theta - \frac{|d_{ij}^A - d_{ij}^B|}{d_{ij}^*} \right) w(d_{ij}^*)$$

Equation 5.5.2

where d_{ij}^* is the average of d_{ij}^A, d_{ij}^B , θ is the similarity threshold, and w is an envelope function. Dali uses the value of θ equal to 0.2. Since pairs in the long distance range are abundant but less discriminative, their contribution is weighted down by the envelope function $w(r) = \exp(-r^2/\alpha^2)$, where $\alpha = 20 \text{ \AA}$, calibrated on the size of a typical domain. Alignments generated using the similarity measure of Equation 5.5.2 are reported, imposing the constraint of strictly sequential alignment. The resulting raw Dali score describing the structural similarity is given by:

$$S(A, B) = \sum_{i \in \text{core}} \sum_{j \in \text{core}} \left(0.2 - \frac{|d_{ij}^A - d_{ij}^B|}{d_{ij}^*} \right) \exp \left(- \frac{d_{ij}^{*2}}{20^2 \text{ \AA}^2} \right)$$

Equation 5.5.3

where values of constants in the equation are explicitly inserted. The core is defined as a set of equivalences between residues in A and B proteins, which is analogous to a sequence alignment.

For a random pairwise comparison the expected Dali score (Equation 5.5.3) increases with the number of residues in the compared proteins. In order to describe the statistical significance of a pairwise comparison score $S(A, B)$ the Dali server uses the Z-score

defined as

$$Z(A,B) = \frac{S(A,B) - m(L)}{0.5 \cdot m(L)}$$

Equation 5.5.4

where the denominator is an estimation of the average standard deviation of scores for various lengths of protein chains. The approximate experimental relation between the mean score (m) and the average length (with $L < 400$)

$$L = \sqrt{L_A L_B}$$

Equation 5.5.5

of two proteins is given by:

$$m(L) \approx 7.95 + 0.71L - 2.59 \cdot 10^{-4} L^2 - 1.92 \cdot 10^{-6} L^3$$

Equation 5.5.6

The Z-score is computed for every possible pair of domains, and the highest value is reported as the Z-score of the protein pair.

Possible domains are determined by the PUU algorithm (parser for Protein Unfolding Units). The algorithm recursively cuts a structure into smaller compact substructures at the weakest interface. A number of postprocessing rules were introduced to supplement numerical criteria. The whole procedure is fully described in the original publication (Holm and Sander, 1995).

Program Parameters

The following parameters are set at the top of the main Perl script. The default values, as used by the Dali server, are indicated. These parameters mainly affect the pruning of search space in the database search.

`$MINLEN=30` Structures with fewer residues are excluded from comparison. Dali was designed to detect similarities at the level of globular domain folding patterns that involve several secondary structure elements. It is not designed to compare conformations of short peptides.

`$MINSSE=2` The Wolf and Parsi methods reduce the complexity of the structural comparison by representing structures (partly) as secondary structure elements. If there are fewer than `$MINSSE` secondary structure elements in the protein, then the Soap method is used.

`$cut0=20.0; $cut1=4.0; $cut2=2.0` The database search by the Dali server uses a set of rules to prune search space after a strong similarity has been found. If a similarity has been found that is above a Z-score equal to `$cut0`, then the search is stopped completely because the query is structurally almost identical to the best hit. If similarities have been found with Z-scores above `$cut1`, then the search list is restricted to the first neighbor shells of all hits. If the best Z-score lies between `$cut1` and `$cut2`, then the search list is restricted to the second neighbor shells of all hits.

`$nbest=1` This parameter controls the number of hits in output. All hits with a Z-score above 2, or at least `$nbest` hits, will be reported.

Comparative Protein Structure Modeling Using Modeller

Functional characterization of a protein sequence is one of the most frequent problems in biology. This task is usually facilitated by an accurate three-dimensional (3-D) structure of the studied protein. In the absence of an experimentally determined structure, comparative or homology modeling often provides a useful 3-D model for a protein that is related to at least one known protein structure (Marti-Renom et al., 2000; Fiser, 2004; Misura and Baker, 2005; Petrey and Honig, 2005; Misura et al., 2006). Comparative modeling predicts the 3-D structure of a given protein sequence (target) based primarily on its alignment to one or more proteins of known structure (templates).

Comparative modeling consists of four main steps (Marti-Renom et al., 2000; Figure 5.6.1): (i) fold assignment, which identifies similarity between the target and at least one

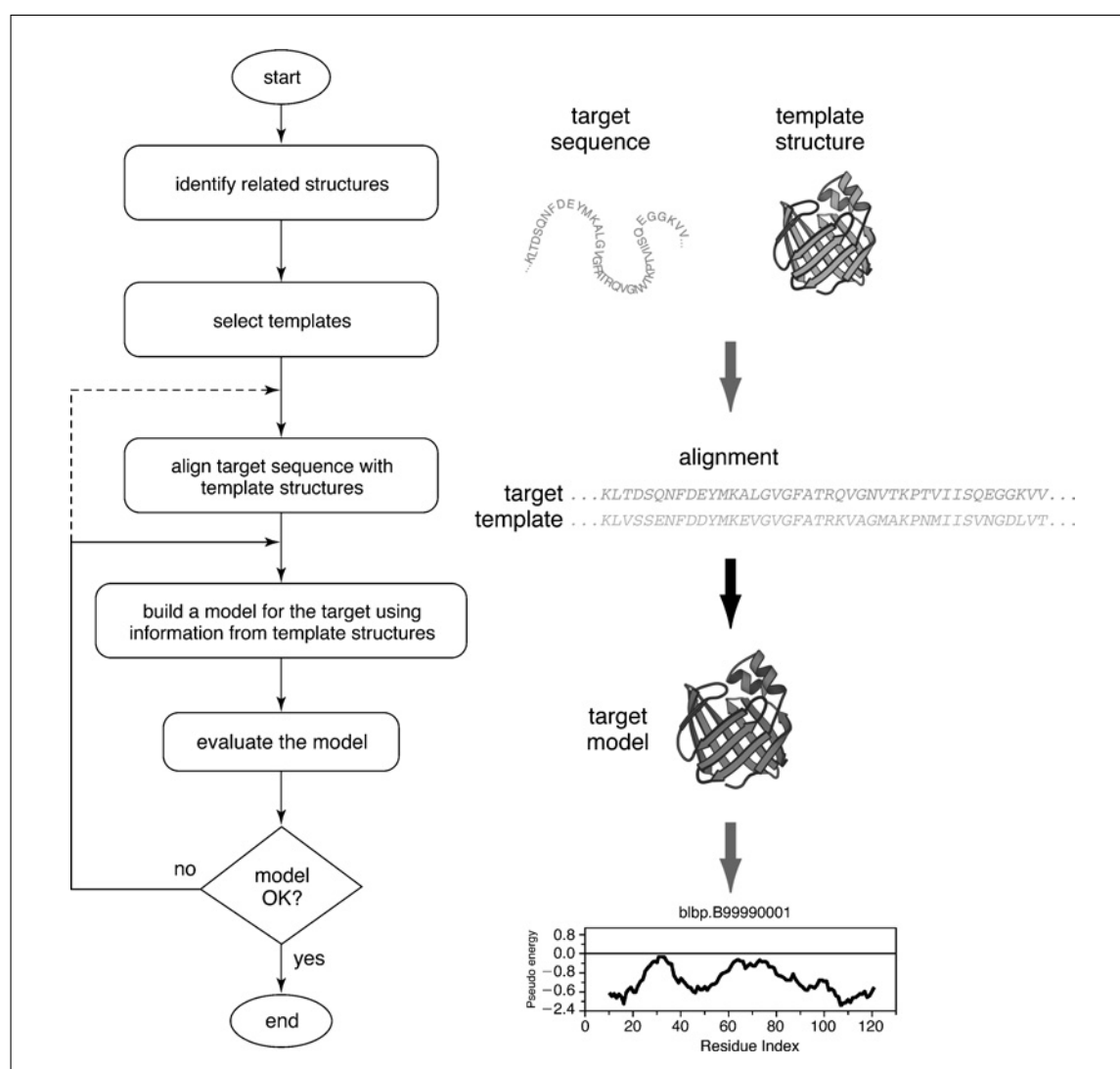


Figure 5.6.1 Steps in comparative protein structure modeling. See text for details. For the color version of this figure go to <http://www.currentprotocols.com>.

Table 5.6.1 Programs and Web Servers Useful in Comparative Protein Structure Modeling

Name	World Wide Web address
Databases	
BALIBASE (Thompson et al., 1999)	http://bips.u-strasbg.fr/en/Products/Databases/BALiBASE/
CATH (Pearl et al., 2005)	http://www.biochem.ucl.ac.uk/bsm/cath/
DBALI (Marti-Renom et al., 2001)	http://www.salilab.org/dbali
GENBANK (Benson et al., 2005)	http://www.ncbi.nlm.nih.gov/Genbank/
GENECENSUS (Lin et al., 2002)	http://bioinfo.mbb.yale.edu/genome/
MODBASE (Pieper et al., 2004)	http://www.salilab.org/modbase/
PDB (UNIT 1.9; Deshpande et al., 2005)	http://www.rcsb.org/pdb/
PFAM (UNIT 2.5; Bateman et al., 2004)	http://www.sanger.ac.uk/Software/Pfam/
SCOP (Andreeva et al., 2004)	http://scop.mrc-lmb.cam.ac.uk/scop/
SWISSPROT (Boeckmann et al., 2003)	http://www.expasy.org
UNIPROT (Bairoch et al., 2005)	http://www.uniprot.org
Template search	
123D (Alexandrov et al., 1996)	http://123d.ncifcrf.gov/
3D PSSM (Kelley et al., 2000)	http://www.sbg.bio.ic.ac.uk/~3dpssm
BLAST (UNIT 3.4; Altschul et al., 1997)	http://www.ncbi.nlm.nih.gov/BLAST/
DALI (UNIT 5.5; Dietmann et al., 2001)	http://www2.ebi.ac.uk/dali/
FASTA (UNIT 3.9; Pearson, 2000)	http://www.ebi.ac.uk/fasta33/
FFAS03 (Jaroszewski et al., 2005)	http://ffas.ljcrf.edu/
PREDICTPROTEIN (Rost and Liu, 2003)	http://cubic.bioc.columbia.edu/predictprotein/
PROSPECTOR (Skolnick and Kihara, 2001)	http://www.bioinformatics.buffalo.edu/new_buffalo/services/threading.html
PSIPRED (McGuffin et al., 2000)	http://bioinf.cs.ucl.ac.uk/psipred/
RAPTOR (Xu et al., 2003)	http://genome.math.uwaterloo.ca/~raptor/
SUPERFAMILY (Gough et al., 2001)	http://supfam.mrc-lmb.cam.ac.uk/SUPERFAMILY/
SAM-T02 (Karplus et al., 2003)	http://www.soe.ucsc.edu/research/compbio/HMM-apps/
SP3 (Zhou and Zhou, 2005)	http://phyzz4.med.buffalo.edu/
SPARKS2 (Zhou and Zhou, 2004)	http://phyzz4.med.buffalo.edu/
THREADER (Jones et al., 1992)	http://bioinf.cs.ucl.ac.uk/threader/threader.html
UCLA-DOE FOLD SERVER (Mallick et al., 2002)	http://fold.doe-mpi.ucla.edu
Target-template alignment	
BCM SERVERF (Worley et al., 1998)	http://searchlauncher.bcm.tmc.edu
BLOCK MAKERF (UNIT 2.2; Henikoff et al., 2000)	http://blocks.fhcrc.org/
CLUSTALW (UNIT 2.3; Thompson et al., 1994)	http://www2.ebi.ac.uk/clustalw/
COMPASS (Sadreyev and Grishin, 2003)	ftp://iole.swmed.edu/pub/compass/

continued

Table 5.6.1 Programs and Web Servers Useful in Comparative Protein Structure Modeling, *continued*

Name	World Wide Web address
Target-template alignment (continued)	
FUGUE (Shi et al., 2001)	http://www-cryst.bioc.cam.ac.uk/fugue
MULTALIN (Corpet, 1988)	http://prodes.toulouse.inra.fr/multalin/
MUSCLE (UNIT 6.9; Edgar, 2004)	http://www.drive5.com/muscle
SALIGN (Eswar et al., 2003)	http://www.salilab.org/modeller
SEA (Ye et al., 2003)	http://ffas.ljcrf.edu/seal
TCOFFEE (UNIT 3.8; Notredame et al., 2000)	http://www.ch.embnet.org/software/TCoffee.html
USC SEQALN (Smith and Waterman, 1981)	http://www-hto.usc.edu/software/seqaln
Modeling	
3D-JIGSAW (Bates et al., 2001)	http://www.bmm.icnet.uk/servers/3djigsaw/
COMPOSER (Sutcliffe et al., 1987a)	http://www.tripos.com
CONGEN (Brucoleri and Karplus, 1990)	http://www.congenomics.com/
ICM (Abagyan and Totrov, 1994)	http://www.molsoft.com
JACKAL (Petrey et al., 2003)	http://trantor.bioc.columbia.edu/programs/jackal/
DISCOVERY STUDIO	http://www.accelrys.com
MODELLER (Sali and Blundell, 1993)	http://www.salilab.org/modeller/
SYBYL	http://www.tripos.com
SCWRL (Canutescu et al., 2003)	http://dunbrack.fccc.edu/SCWRL3.php
SNPWEB (Eswar et al., 2003)	http://salilab.org/snpweb
SWISS-MODEL (Schwede et al., 2003)	http://www.expasy.org/swissmod
WHAT IF (Vriend, 1990)	http://www.cmbi.kun.nl/whatif/
Prediction of model errors	
ANOLEA (Melo and Feytmans, 1998)	http://protein.bio.puc.cl/cardex/servers/
AQUA (Laskowski et al., 1996)	http://urchin.bmr.b.wisc.edu/~jürgen/aqua/
BIOTECH (Laskowski et al., 1998)	http://biotech.embl-heidelberg.de:8400
ERRAT (Colovos and Yeates, 1993)	http://www.doe-mpi.ucla.edu/Services/ERRAT/
PROCHECK (Laskowski et al., 1993)	http://www.biochem.ucl.ac.uk/~roman/procheck/procheck.html
PROSAIL (Sippl, 1993)	http://www.came.sbg.ac.at
PROVE (Pontius et al., 1996)	http://www.ucmb.ulb.ac.be/UCMB/PROVE
SQUID (Oldfield, 1992)	http://www.ysbl.york.ac.uk/~oldfield/squid/
VERIFY3D (Luthy et al., 1992)	http://www.doe-mpi.ucla.edu/Services/Verify_3D/
WHATCHECK (Hooft et al., 1996)	http://www.cmbi.kun.nl/gv/whatcheck/
Methods evaluation	
CAFASP (Fischer et al., 2001)	http://cafasp.bioinfo.pl
CASP (Moult et al., 2003)	http://predictioncenter.llnl.gov
CASA (Kahsay et al., 2002)	http://capb.dbi.udel.edu/casa
EVA (Koh et al., 2003)	http://cubic.bioc.columbia.edu/eval
LIVEBENCH (Bujnicki et al., 2001)	http://bioinfo.pl/LiveBench/

known template structure; (ii) alignment of the target sequence and the template(s); (iii) building a model based on the alignment with the chosen template(s); and (iv) predicting model errors.

There are several computer programs and Web servers that automate the comparative modeling process (Table 5.6.1). The accuracy of the models calculated by many of these servers is evaluated by EVA-CM (Eyrich et al., 2001), LiveBench (Bujnicki et al., 2001), and the biannual CASP (Critical Assessment of Techniques for Proteins Structure Prediction; Moult, 2005; Moult et al., 2005) and CAFASP (Critical Assessment of Fully Automated Structure Prediction) experiments (Rychlewski and Fischer, 2005; Fischer, 2006).

While automation makes comparative modeling accessible to both experts and nonspecialists, manual intervention is generally still needed to maximize the accuracy of the models in the difficult cases. A number of resources useful in comparative modeling are listed in Table 5.6.1.

This unit describes how to calculate comparative models using the program MODELLER (Basic Protocol). The Basic Protocol goes on to discuss all four steps of comparative modeling (Figure 5.6.1), frequently observed errors, and some applications. The Support Protocol describes how to download and install MODELLER.

BASIC PROTOCOL

MODELING LACTATE DEHYDROGENASE FROM *TRICHOMONAS VAGINALIS* (TvLDH) BASED ON A SINGLE TEMPLATE USING MODELLER

MODELLER is a computer program for comparative protein structure modeling (Sali and Blundell, 1993; Fiser et al., 2000). In the simplest case, the input is an alignment of a sequence to be modeled with the template structures, the atomic coordinates of the templates, and a simple script file. MODELLER then automatically calculates a model containing all non-hydrogen atoms, within minutes on a Pentium processor and with no user intervention. Apart from model building, MODELLER can perform additional auxiliary tasks, including fold assignment (Eswar, 2005), alignment of two protein sequences or their profiles (Marti-Renom et al., 2004), multiple alignment of protein sequences and/or structures (Madhusudhan et al., 2006), calculation of phylogenetic trees, and de novo modeling of loops in protein structures (Fiser et al., 2000).

NOTE: Further help for all the described commands and parameters may be obtained from the MODELLER Web site (see Internet Resources).

Necessary Resources

Hardware

A computer running RedHat Linux (PC, Opteron, EM64T/Xeon64, or Itanium 2 systems) or other version of Linux/Unix (x86/x86_64/IA64 Linux, Sun, SGI, Alpha, AIX), Apple Mac OSX (PowerPC), or Microsoft Windows 98/2000/XP

Software

The MODELLER 8v2 program, downloaded and installed from
http://salilab.org/modeller/download_installation.html (see Support Protocol)

Files

All files required to complete this protocol can be downloaded from
<http://salilab.org/modeller/tutorial/basic-example.tar.gz> (Unix/Linux) or
<http://salilab.org/modeller/tutorial/basic-example.zip> (Windows)

```

>P1;TvLDH
sequence:TvLDH:::::0.00: 0.00
MSEAAHVLIITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLMLDQNRAYYEVAASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFKKI
GHRWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSPFCNVCKEKGKIHVV
EGFKVNDWLREKLDLFEKDLFHEKEIALNHLAQGG*

```

Figure 5.6.2 File TvLDH.ali. Sequence file in PIR format.

Background to TvLDH

A novel gene for lactate dehydrogenase (LDH) was identified from the genomic sequence of *Trichomonas vaginalis* (TvLDH). The corresponding protein had higher sequence similarity to the malate dehydrogenase of the same species (TvMDH) than to any other LDH. The authors hypothesized that TvLDH arose from TvMDH by convergent evolution relatively recently (Wu et al., 1999). Comparative models were constructed for TvLDH and TvMDH to study the sequences in a structural context and to suggest site-directed mutagenesis experiments to elucidate changes in enzymatic specificity in this apparent case of convergent evolution. The native and mutated enzymes were subsequently expressed and their activities compared (Wu et al., 1999).

Searching structures related to TvLDH

Conversion of sequence to PIR file format

It is first necessary to convert the target TvLDH sequence into a format that is readable by MODELLER (file TvLDH.ali; Fig. 5.6.2). MODELLER uses the PIR format to read and write sequences and alignments. The first line of the PIR-formatted sequence consists of >P1; followed by the identifier of the sequence. In this example, the sequence is identified by the code TvLDH. The second line, consisting of ten fields separated by colons, usually contains details about the structure, if any. In the case of sequences with no structural information, only two of these fields are used: the first field should be sequence (indicating that the file contains a sequence without a known structure) and the second should contain the model file name (TvLDH in this case). The rest of the file contains the sequence of TvLDH, with an asterisk (*) marking its end. The standard uppercase single-letter amino acid codes are used to represent the sequence.

Searching for suitable template structures

A search for potentially related sequences of known structure can be performed using the `profile.build()` command of MODELLER (file `build_profile.py`). The command uses the local dynamic programming algorithm to identify related sequences (Smith and Waterman, 1981; Eswar, 2005). In the simplest case, the command takes as input the target sequence and a database of sequences of known structure (file `pdb_95.pir`) and returns a set of statistically significant alignments. The input script file for the command is shown in Figure 5.6.3.

The script, `build_profile.py`, does the following:

1. Initializes the “environment” for this modeling run by creating a new `environ` object (called `env` here). Almost all MODELLER scripts require this step, as the new object is needed to build most other useful objects.
2. Creates a new `sequence_db` object, calling it `sdb`, which is used to contain large databases of protein sequences.

```

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.read(seq_database_file='pdb_95.pir', seq_database_format='PIR',
        chains_list='ALL', minmax_db_seq_len=(30, 4000), clean_sequences=True)

sdb.write(seq_database_file='pdb_95.bin', seq_database_format='BINARY',
        chains_list='ALL')

sdb.read(seq_database_file='pdb_95.bin', seq_database_format='BINARY',
        chains_list='ALL')

aln = alignment(env)
aln.append(file='TvLDH.ali', alignment_format='PIR', align_codes='ALL')

prf = aln.to_profile()

prf.build(sdb, matrix_offset=-450, rr_file='${LIB}/blosum62.sim.mat',
        gap_penalties_ld=(-500, -50), n_prof_iterations=1,
        check_profile=False, max_aln_evalue=0.01)

prf.write(file='build_profile.prf')

aln = prf.to_alignment()

aln.write(file='build_profile.ali', alignment_format='PIR')

```

Figure 5.6.3 File `build_profile.py`. Input script file that searches for templates against a database of nonredundant PDB sequences.

3. Reads a file, in text format, containing nonredundant PDB sequences, into the `sdb` database. The sequences can be found in the file `pdb_95.pir`. This file is also in the PIR format. Each sequence in this file is representative of a group of PDB sequences that share 95% or more sequence identity to each other and have less than 30 residues or 30% sequence length difference.
4. Writes a binary machine-independent file containing all sequences read in the previous step.
5. Reads the binary format file back in for faster execution.
6. Creates a new “alignment” object (`aln`), reads the target sequence `TvLDH` from the file `TvLDH.ali`, and converts it to a profile object (`prf`). Profiles contain similar information to alignments, but are more compact and better for sequence database searching.
7. `prf.build()` searches the sequence database (`sdb`) with the target profile (`prf`). Matches from the sequence database are added to the profile.
8. `prf.write()` writes a new profile containing the target sequence and its homologs into the specified output file (file `build_profile.prf`; Fig. 5.6.4). The equivalent information is also written out in standard alignment format.

The `profile.build()` command has many options (see Internet Resources for MODELLER Web site). In this example, `rr_file` is set to use the BLOSUM62 similarity matrix (file `blosum62.sim.mat` provided in the MODELLER distribution). Accordingly, the parameters `matrix_offset` and `gap_penalties_ld` are set to the appropriate values for the BLOSUM62 matrix. For this example, only one search iteration is run, by setting the parameter `n_prof_iterations` equal to 1. Thus, there is no need to check the profile for deviation (`check_profile` set to `False`). Finally,

```

# Number of sequences :      30
# Length of profile   :      335
# N_PROF_ITERATIONS  :        1
# GAP_PENALTIES_1D   :     -900.0  -50.0
# MATRIX_OFFSET      :        0.0
# RR_FILE             :  ${MODINSTALL8v0}/modlib//asl.sim.mat

  1  TvLDH                      S      0  335   1  335   0   0   0   0.  0.0
  2  1a5z                      X      1  312  75  242  63  229  164  28.  0.83E-08
  3  1b8pA                     X      1  327   7  331   6  325  316  42.  0.0
  4  1bdmA                     X      1  318   1  325   1  310  309  45.  0.0
  5  1t2dA                     X      1  315   5  256   4  250  238  25.  0.66E-04
  6  1civA                     X      1  374   6  334  33  358  325  35.  0.0
  7  2cmd                      X      1  312   7  320   3  303  289  27.  0.16E-05
  8  1o6zA                     X      1  303   7  320   3  287  278  26.  0.27E-05
  9  1ur5A                     X      1  299  13  191   9  171  158  31.  0.25E-02
 10  1guzA                     X      1  305  13  301   8  280  265  25.  0.28E-08
 11  1gv0A                     X      1  301  13  323   8  289  274  26.  0.28E-04
 12  1hyeA                     X      1  307   7  191   3  183  173  29.  0.14E-07
 13  1iOzA                     X      1  332  85  300  94  304  207  25.  0.66E-05
 14  1i10A                     X      1  331  85  295  93  298  196  26.  0.86E-05
 15  1idnA                     X      1  316  78  298  73  301  214  26.  0.19E-03
 16  6ldh                      X      1  329  47  301  56  302  244  23.  0.17E-02
 17  2ldx                      X      1  331  66  306  67  306  227  26.  0.25E-04
 18  5ldh                      X      1  333  85  300  94  304  207  26.  0.30E-05
 19  9ldtA                     X      1  331  85  301  93  304  207  26.  0.10E-05
 20  111c                      X      1  321  64  239  53  234  164  26.  0.20E-03
 21  111dA                     X      1  313  13  242   9  233  216  31.  0.31E-07
 22  5mdhA                     X      1  333   2  332   1  331  328  44.  0.0
 23  7mdhA                     X      1  351   6  334  14  339  325  34.  0.0
 24  1ml.dA                   X      1  313   5  198   1  189  183  26.  0.13E-05
 25  1oc4A                     X      1  315   5  191   4  186  174  28.  0.18E-04
 26  1ojuA                     X      1  294  78  320  68  285  218  28.  0.43E-05
 27  1pzgA                     X      1  327  74  191  71  190  114  30.  0.16E-06
 28  1smkA                     X      1  313   7  202   4  198  188  34.  0.0
 29  1sovA                     X      1  316  81  256  76  248  160  27.  0.93E-03
 30  1y63A                     X      1  289  77  191  58  167  109  33.  0.32E-05

```

Figure 5.6.4 An excerpt from the file `build_profile.prf`. The aligned sequences have been removed for convenience.

the parameter `max_aln_evalue` is set to 0.01, indicating that only sequences with *E*-values smaller than or equal to 0.01 will be included in the output.

Execute the script using the command `mod8v2 build_profile.py`. At the end of the execution, a log file is created (`build_profile.log`). MODELLER always produces a log file. Errors and warnings in log files can be found by searching for the `.E>` and `.W>` strings, respectively.

Selecting a template

An extract (omitting the aligned sequences) from the file `build_profile.prf` is shown in Figure 5.6.4. The first six commented lines indicate the input parameters used in MODELLER to create the alignments. Subsequent lines correspond to the detected similarities by `profile.build()`. The most important columns in the output are the second, tenth, eleventh, and twelfth columns. The second column reports the code of the PDB sequence that was aligned to the target sequence. The eleventh column reports the percentage sequence identities between TvLDH and the PDB sequence normalized by the length of the alignment (indicated in the tenth column). In general, a sequence identity value above ~25% indicates a potential template, unless the alignment is too short (i.e., <100 residues). A better measure of the significance of the alignment is given in the twelfth column by the *E*-value of the alignment (lower the *E*-value the better).

In this example, six PDB sequences show very significant similarities to the query sequence, with *E*-values equal to 0. As expected, all the hits correspond to malate dehydrogenases (1bdm:A, 5mdh:A, 1b8p:A, 1civ:A, 7mdh:A, and 1smk:A). To select the appropriate template for the target sequence, the `alignment.compare_structures()`

```

env = environ()
aln = alignment(env, file='${LIB}/CHAINS_all.seq',
               align_codes=('1b8pA', '1bdmA', '1civA', '5mdhA', '7mdhA', '1smkA'))
aln.malign()
aln.malign3d()
aln.compare_structures()
aln.id_table()
env.dendrogram()

```

Figure 5.6.5 Script file `compare.py`.

command will first be used to assess the sequence and structure similarity between the six possible templates (file `compare.py`; Fig. 5.6.5).

In `compare.py`, the alignment object `aln` is created and MODELLER is instructed to read into it the protein sequences and information about their PDB files. By default, all sequences from the provided file are read in, but in this case, the user should restrict it to the selected six templates by specifying their `align_codes`. The command `malign()` calculates their multiple sequence alignment, which is subsequently used as a starting point for creating a multiple structure alignment by `malign3d()`. Based on this structural alignment, the `compare_structures()` command calculates the RMS and DRMS deviations between atomic positions and distances, differences between the main-chain and side-chain dihedral angles, percentage sequence identities, and several other measures. Finally, the `id_table()` command writes a file (`family.mat`) with pairwise sequence distances that can be used as input to the `dendrogram()` command (or the clustering programs in the PHYLIP package; Felsenstein, 1989). `dendrogram()` calculates a clustering tree from the input matrix of pairwise distances, which helps visualizing differences among the template candidates. Excerpts from the log file (`compare.log`) are shown in Figure 5.6.6.

The objective of this step is to select the most appropriate single template structure from all the possible templates. The dendrogram in Figure 5.6.6 shows that `1civ:A` and `7mdh:A` are almost identical, both in terms of sequence and structure. However, `7mdh:A` has a better crystallographic resolution than `1civ:A` (2.4 Å versus 2.8 Å). From the second group of similar structures (`5mdh:A`, `1bdm:A`, and `1b8p:A`), `1bdm:A` has the best resolution (1.8 Å). `1smk:A` is most structurally divergent among the possible templates. However, it is also the one with the lowest sequence identity (34%) to the target sequence (`build_profile.prf`). `1bdm:A` is finally picked over `7mdh:A` as the final template because of its higher overall sequence identity to the target sequence (45%).

Aligning TvLDH with the template

One way to align the sequence of TvLDH with the structure of `1bdm:A` is to use the `align2d()` command in MODELLER (Madhusudhan et al., 2006). Although `align2d()` is based on a dynamic programming algorithm (Needleman and Wunsch, 1970), it is different from standard sequence-sequence alignment methods because it takes into account structural information from the template when constructing an alignment. This task is achieved through a variable gap penalty function that tends to place gaps in solvent-exposed and curved regions, outside secondary structure segments, and between two positions that are close in space. In the current example, the target-template similarity is so high that almost any alignment method with reasonable parameters will result in the same alignment.

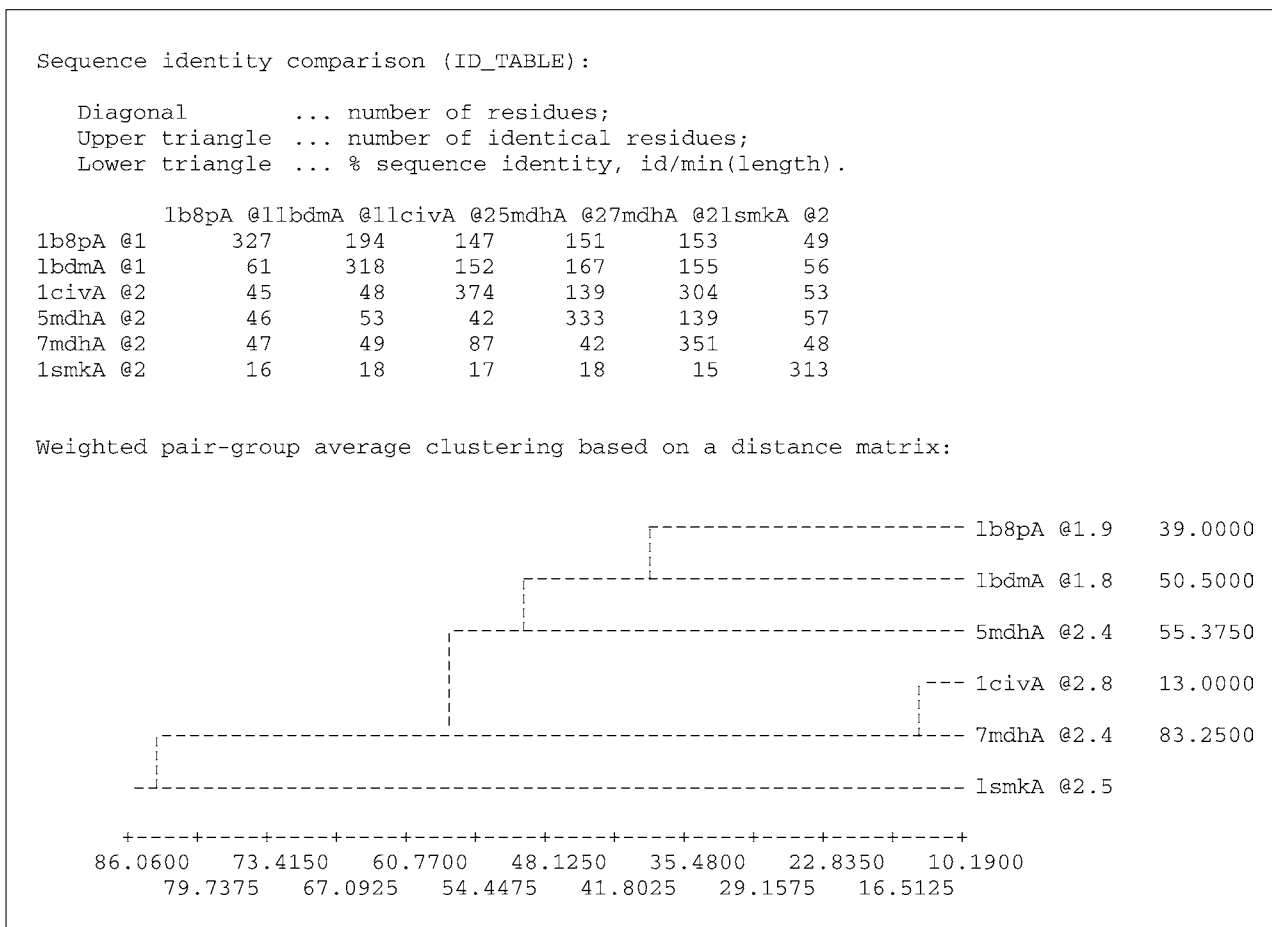


Figure 5.6.6 Excerpts from the log file `compare.log`.

```

env = environ()
aln = alignment(env)

mdl = model(env, file='lbdm', model_segment=('FIRST:A', 'LAST:A'))
aln.append_model(mdl, align_codes='lbdmA', atom_files='lbdm.pdb')
aln.append(file='TvLDH.ali', align_codes='TvLDH')

aln.align2d()
aln.write(file='TvLDH-lbdmA.ali', alignment_format='PIR')
aln.write(file='TvLDH-lbdmA.pap', alignment_format='PAP')

```

Figure 5.6.7 The script file `align2d.py`, used to align the target sequence against the template structure.

The MODELLER script shown in Figure 5.6.7 aligns the TvLDH sequence in file `TvLDH.ali` with the lbdm:A structure in the PDB file `lbdm.pdb` (file `align2d.py`). In the first line of the script, an empty alignment object `aln`, and a new model object `mdl`, into which the chain A of the lbmd structure is read, are created. `append_model()` transfers the PDB sequence of this model to `aln` and assigns it the name of `lbdmA` (`align_codes`). The TvLDH sequence, from file `TvLDH.ali`, is then added to `aln` using `append()`. The `align2d()` command aligns the two sequences and the alignment is written out in two formats, PIR (`TvLDH-lbdmA.ali`) and PAP (`TvLDH-lbdmA.pap`). The PIR format is used by MODELLER in the subsequent model-building stage, while the PAP alignment format is easier to inspect visually. In the PAP format, all identical positions are marked with a `*` (file `TvLDH-lbdmA.pap`; Fig. 5.6.8). Due to the high target-template similarity, there are only a few gaps in the alignment.

```

aln.pos      10      20      30      40      50      60
ThdmA      MKAPVRVAVTGAAGQIGYSLLFRIAAGEMLGKDQPVILQLLEIPQAMKALEGVVMELEDCAFPLLAGL
TvLDH      MSEAAHVLTGAAGQIGYILSHWIASGELYG-DRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGF
_consrvd   *      *      *      *      *      *      *      *      *      *      *      *      *      *      *      *

aln.p      70      80      90      100      110      120      130
ThdmA      EATDDPDVAFKADADYALLVGAAPRL -----QVNGKIFTEQGRALAEVAKKDVKVLVVGNPANTN
TvLDH      VATTDPKAAFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTN
_consrvd   ** **      *      *      *      *      *      *      *      *      *      *      *      *      *      *

aln.pos     140      150      160      170      180      190      200
lbdmA      ALIAYKNAPGLNPRNFTAMTRLDHNRAKAQLAKKTGTGVDRIRRM TVWGNHSSIMFPDLFHA EVD---
TvLDH      CEIAMLHAKNLKPNFSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEG
_consrvd   **      *      *      *      *      *      *      *      *      *      *      *      *      *      *      *

aln.pos     210      220      230      240      250      260      270
ThdmA      -GRP ALELVDM EWYEKVF IPTVAQRGA AIIQARGASSAASAANA AIEHIRD WALGTPEGDWVSM AVPS
TvLDH      KTQKVVDVLDHDYVFD TFFKKIGHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPV
_consrvd   *      *      *      *      *      *      *      *      *      *      *      *      *      *      *      *

aln.pos     280      290      300      310      320      330
ThdmA      Q--GEYGIPEGIVYSFPVTAK-DGAYRVVEGLEINEFARKRMEITAQELLDEMEQVKAL--GLI
TvLDH      PEGNPYGIKPGVVFSFPCNV DKEGIHVVVEGFKVNDWLREKLDFT EKDLFHEKEIALNHLAQGG
_consrvd   ***      *      *      *      *      *      *      *      *      *      *      *      *      *      *

```

Figure 5.6.8 The alignment between sequences TvLDH and lbdmA, in the MODELLER PAP format. File TvLDH-lbdmA.pap.

```

from modeller.automodel import *

env = environ()
a = automodel(env, alnfile='TvLDH-lbdmA.ali',
               knowns='lbdmA', sequence='TvLDH')
a.starting_model = 1
a.ending_model = 5
a.make()

```

Figure 5.6.9 Script file, model-single.py, that generates five models.

Model building

Once a target-template alignment is constructed, MODELLER calculates a 3-D model of the target completely automatically, using its automodel class. The script in Figure 5.6.9 will generate five different models of TvLDH based on the lbdm:A template structure and the alignment in file TvLDH-lbdmA.ali (file model-single.py).

The first line (Fig. 5.6.9) loads the automodel class and prepares it for use. An automodel object is then created and called “a,” and parameters are set to guide the model-building procedure. alnfile names the file that contains the target-template alignment in the PIR format. knowns defines the known template structure(s) in alnfile (TvLDH-lbdmA.ali) and sequence defines the code of the target sequence. starting_model and ending_model define the number of models that are calculated (their indices will run from 1 to 5). The last line in the file calls the make method that actually calculates the models. The most important output files are model-single.log, which reports warnings, errors and other useful information including the input restraints used for modeling that remain violated in the final model, and TvLDH.B9999000 [1-5].pdb, which contain the coordinates of the five produced models, in the PDB format. The models can be viewed by any program that reads the PDB format, such as Chimera (<http://www.cgl.ucsf.edu/chimera/>) or RasMol (<http://www.rasmol.org>).

```

from modeller.automodel import *

log.verbose()
env = environ()
env.libs.topology.read(file='${LIB}/top_heav.lib')
env.libs.parameters.read(file='${LIB}/par.lib')

env.io.atom_files_directory = './../atom-files'

mdl = model(env)
mdl.read(file='TvLDH.B99990001.pdb')
aln = alignment(env)
code = "TvLDH"

aln.append_model(mdl, atom_files='TvLDH.B99990001.pdb', align_codes=code)
aln.append_model(mdl, atom_files='TvLDH.B99990001.pdb', align_codes=code+'-ini')
mdl.generate_topology(aln, sequence=code+'-ini')
mdl.transfer_xyz(aln)

mdl.assess_dope(output='ENERGY_PROFILE NO_REPORT', file='TvLDH.profile',
               normalize_profile=True, smoothing_window=15)

```

Figure 5.6.10 File `evaluate_model.py`, used to generate a pseudo-energy profile for the model.

Evaluating a model

If several models are calculated for the same target, the best model can be selected by picking the model with the lowest value of the MODELLER objective function, which is reported in the second line of the model PDB file. In this example, the first model (TvLDH.B99990001.pdb) has the lowest objective function. The value of the objective function in MODELLER is not an absolute measure, in the sense that it can only be used to rank models calculated from the same alignment.

Once a final model is selected, there are many ways to assess it. In this example, the DOPE potential in MODELLER is used to evaluate the fold of the selected model. Links to other programs for model assessment can be found in Table 5.6.1. However, before any external evaluation of the model, one should check the log file from the modeling run for runtime errors (`model-single.log`) and restraint violations (see the MODELLER manual for details).

The script, `evaluate_model.py` (Fig. 5.6.10) evaluates the model with the DOPE potential. In this script, sequence is first transferred (using `append_model()`), and then the atomic coordinates of the PDB file are transferred (using `transfer_xyz()`), to a model object, `mdl`. This is necessary for MODELLER to correctly calculate the energy, and additionally allows for the possibility of the PDB file having atoms in a nonstandard order, or having different subsets of atoms (e.g., all atoms including hydrogens, while MODELLER uses only heavy atoms, or vice versa). The DOPE energy is then calculated using `assess_dope()`. An energy profile is additionally requested, smoothed over a 15-residue window, and normalized by the number of restraints acting on each residue. This profile is written to a file `TvLDH.profile`, which can be used as input to a graphing program such as GNU PLOT.

Similarly, `evaluate_model.py` calculates a profile for the template structure. A comparison of the two profiles is shown in Figure 5.6.11. It can be seen that the DOPE score profile shows clear differences between the two profiles for the long active-site loop between residues 90 and 100 and the long helices at the C-terminal end of the target sequence. This long loop interacts with region 220 to 250, which forms the other half of the active site. This latter region is well resolved in both the template and the target structure. However, probably due to the unfavorable nonbonded interactions with the 90 to 100

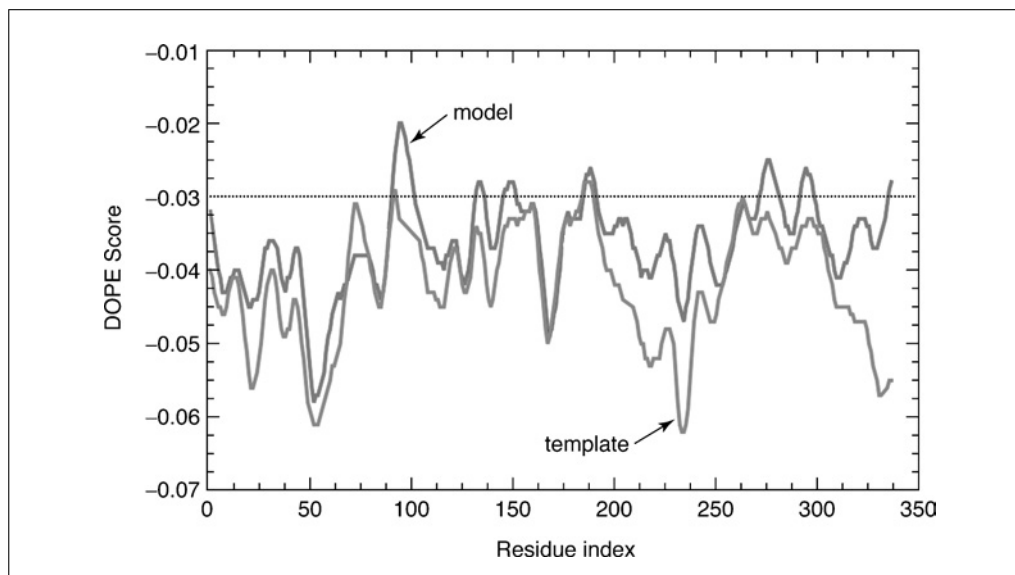


Figure 5.6.11 A comparison of the pseudo-energy profiles of the model (red) and the template (green) structures. For the color version of this figure go to <http://www.currentprotocols.com>.

region, it is reported to be of high energy by DOPE. It is to be noted that a region of high energy indicated by DOPE may not always necessarily indicate actual error, especially when it highlights an active site or a protein-protein interface. However, in this case, the same active-site loops have a better profile in the template structure, which strengthens the argument that the model is probably incorrect in the active-site region. Resolution of such problems is beyond the scope of this unit, but is described in a more advanced modeling tutorial available at <http://salilab.org/modeller/tutorial/advanced.html>.

SUPPORT PROTOCOL

OBTAINING AND INSTALLING MODELLER

MODELLER is written in Fortran 90 and uses Python for its control language. All input scripts to MODELLER are, hence, Python scripts. While knowledge of Python is not necessary to run MODELLER, it can be useful in performing more advanced tasks. Pre-compiled binaries for MODELLER can be downloaded from <http://salilab.org/modeller>.

Necessary Resources

Hardware

A computer running RedHat Linux (PC, Opteron, EM64T/Xeon64 or Itanium 2 systems) or other version of Linux/Unix (x86/x86_64/IA64 Linux, Sun, SGI, Alpha, AIX), Apple Mac OS X (PowerPC), or Microsoft Windows 98/2000/XP

Software

An up-to-date Internet browser, such as Internet Explorer (<http://www.microsoft.com/ie>); Netscape (<http://browser.netscape.com>); Firefox (<http://www.mozilla.org/firefox>); or Safari (<http://www.apple.com/safari>)

Installation

The steps involved in installing MODELLER on a computer depend on its operating system. The following procedure describes the steps for installing MODELLER on a generic x86 PC running any Unix/Linux operating system. The procedures for other operating systems differ slightly. Detailed instructions for installing MODELLER on machines running other operating systems can be found at <http://salilab.org/modeller/release.html>.

1. Point browser to http://salilab.org/modeller/download_installation.html.
2. On the page that appears, download the distribution by clicking on the link entitled “Other Linux/Unix” under “Available downloads...”.
3. A valid license key, distributed free of cost to academic users, is required to use MODELLER. To obtain a key, go to the URL <http://salilab.org/modeller/registration.html>, fill in the simple form at the bottom of the page, and read and accept the license agreement. The key will be E-mailed to the address provided.
4. Open a terminal or console and change to the directory containing the downloaded distribution. The distributed file is a compressed archive file called `modeller-8v2.tar.gz`.
5. Unpack the downloaded file with the following commands:

```
gunzip modeller-8v2.tar.gz
```

```
tar -xvf modeller-8v2.tar
```

6. The files needed for the installation can be found in a newly created directory called `modeller-8v2`. Move into that directory and start the installation with the following commands:
- ```
cd modeller-8v2
```
- ```
./Install
```
7. The installation script will prompt the user with several questions and suggest default answers. To accept the default answers, press the Enter key. The various prompts are briefly discussed below:

- a. For the prompt below, choose the appropriate combination of the machine architecture and operating system. For this example, choose the default answer by pressing the Enter key.

The currently supported architectures are as follows:

- 1) Linux x86 PC (e.g., RedHat, SuSe).
- 2) SUN Inc. Solaris workstation.
- 3) Silicon Graphics Inc. IRIX workstation.
- 4) DEC Inc. Alpha OSF/1 workstation.
- 5) IBM AIX OS.
- 6) Apple Mac OS X 10.3.x (Panther).
- 7) Itanium 2 box (Linux).
- 8) AMD64 (Opteron) or EM64T (Xeon64) box (Linux).
- 9) Alternative Linux x86 PC binary (e.g., for FreeBSD).

Select the type of your computer from the list above
[1]:

- b. For the prompt below, tell the installer where to install the MODELLER executables. The default choice will place it in the directory indicated, but any directory to which the user has write permissions may be specified.

Full directory name for the installed MODELLER8v2
[<YOUR-HOME-DIRECTORY>/bin/modeller8v2]:

- c. For the prompt below, enter the MODELLER license key obtained in step 3.
KEY_MODELLER8v2, obtained from our academic
license server at <http://salilab.org/modeller/registration.shtml>:

8. The installer will now confirm the answers to the above prompts. Press Enter to begin the installation. The mod8v2 script installed in the chosen directory can now be used to invoke MODELLER.

Other resources

9. The MODELLER Web site provides links to several additional resources that can supplement the tutorial provided in this unit, as follows.
 - a. News about the latest MODELLER releases can be found at <http://salilab.org/modeller/news.html>.
 - b. There is a discussion forum, operated through a mailing list, devoted to providing tips, tricks, and practical help in using MODELLER. Users can subscribe to the mailing list at http://salilab.org/modeller/discussion_forum.html. Users can also browse through or search the archived messages of the mailing list.
 - c. The documentation section of the web page contains links to Frequently Asked Questions (FAQ; <http://salilab.org/modeller/FAQ.html>), tutorial examples (<http://salilab.org/modeller/tutorial>), an online version of the manual (<http://salilab.org/modeller/manual>), and user-editable Wiki pages (<http://salilab.org/modeller/wiki/>) to exchange tips, scripts, and examples.

COMMENTARY

Background Information

As stated earlier, comparative modeling consists of four main steps: fold assignment, target-template alignment, model building and model evaluation (Marti-Renom et al., 2000; Fig. 5.6.1).

Fold assignment and target-template alignment

Although fold assignment and sequence-structure alignment are logically two distinct steps in the process of comparative modeling, in practice, almost all fold-assignment methods also provide sequence-structure alignments. In the past, fold-assignment methods were optimized for better sensitivity in detecting remotely related homologs, often at the cost of alignment accuracy. However, recent methods simultaneously optimize both the sensitivity and alignment accuracy. Therefore, in the following discussion, fold assignment and sequence-structure alignment will be treated as a single procedure, explaining the differences as needed.

Fold assignment

The primary requirement for comparative modeling is the identification of one or more known template structures with detectable similarity to the target sequence. The identification of suitable templates is achieved by scanning structure databases, such as PDB (Deshpande et al., 2005), SCOP (Andreeva et al., 2004), DALI, *UNIT 5.5* (Dietmann et al., 2001), and CATH (Pearl et al., 2005), with the target sequence as the query. The detected

similarity is usually quantified in terms of sequence identity or statistical measures such as *E*-value or *z*-score, depending on the method used.

Three regimes of the sequence-structure relationship

The sequence-structure relationship can be subdivided into three different regimes in the sequence similarity spectrum: (i) the easily detected relationships, characterized by >30% sequence identity; (ii) the “twilight zone” (Rost, 1999), corresponding to relationships with statistically significant sequence similarity, with identities in the 10% to 30% range; and (iii) the “midnight zone” (Rost, 1999), corresponding to statistically insignificant sequence similarity.

Pairwise sequence alignment methods

For closely related protein sequences with identities higher than 30% to 40%, the alignments produced by all methods are almost always largely correct. The quickest way to search for suitable templates in this regime is to use simple pairwise sequence alignment methods such as SSEARCH (Pearson, 1994), BLAST (Altschul et al., 1997), and FASTA (Pearson, 1994). Brenner et al. (1998) showed that these methods detect only ~18% of the homologous pairs at less than 40% sequence identity, while they identify more than 90% of the relationships when sequence identity is between 30% and 40% (Brenner et al., 1998). Another benchmark, based on 200 reference structural alignments with 0% to 40%

sequence identity, indicated that BLAST is able to correctly align only 26% of the residue positions (Sauder et al., 2000).

Profile-sequence alignment methods

The sensitivity of the search and accuracy of the alignment become progressively difficult as the relationships move into the twilight zone (Saqi et al., 1998; Rost, 1999). A significant improvement in this area was the introduction of profile methods by Gribskov et al. (1987). The profile of a sequence is derived from a multiple sequence alignment and specifies residue-type occurrences for each alignment position. The information in a multiple sequence alignment is most often encoded as either a position-specific scoring matrix (PSSM; Henikoff and Henikoff, 1994, 1996; Altschul et al., 1997) or as a Hidden Markov Model (HMM; Krogh et al., 1994; Eddy, 1998). In order to identify suitable templates for comparative modeling, the profile of the target sequence is used to search against a database of template sequences. The profile-sequence methods are more sensitive in detecting related structures in the twilight zone than the pairwise sequence-based methods; they detect approximately twice the number of homologs under 40% sequence identity (Park et al., 1998; Lindahl and Elofsson, 2000; Sauder et al., 2000). The resulting profile-sequence alignments correctly align approximately 43% to 48% of residues in the 0% to 40% sequence identity range (Sauder et al., 2000; Marti-Renom et al., 2004); this number is almost twice as large as that of the pairwise sequence methods. Frequently used programs for profile-sequence alignment are PSI-BLAST (Altschul et al., 1997), SAM (Karplus et al., 1998), HMMER (Eddy, 1998), and BUILD.PROFILE (Eswar, 2005).

Profile-profile alignment methods

As a natural extension, the profile-sequence alignment methods have led to profile-profile alignment methods that search for suitable template structures by scanning the profile of the target sequence against a database of template profiles as opposed to a database of template sequences. These methods have proven to include the most sensitive and accurate fold assignment and alignment protocols to date (Edgar and Sjolander, 2004; Marti-Renom et al., 2004; Ohlson et al., 2004; Wang and Dunbrack, 2004). Profile-profile methods detect ~28% more relationships at the superfamily level and improve the alignment accuracy for 15% to 20%, compared to profile-sequence methods (Marti-Renom et al., 2004; Zhou and

Zhou, 2005). There are a number of variants of profile-profile alignment methods that differ in the scoring functions they use (Petrokovski, 1996; Rychlewski et al., 1998; Yona and Levitt, 2002; Panchenko, 2003; Sadreyev and Grishin, 2003; von Ohlsen et al., 2003; Edgar and Sjolander, 2004; Marti-Renom et al., 2004; Zhou and Zhou, 2005). However, several analyses have shown that the overall performances of these methods are comparable (Edgar and Sjolander, 2004; Marti-Renom et al., 2004; Ohlson et al., 2004; Wang and Dunbrack, 2004). Some of the programs that can be used to detect suitable templates are FFAS (Jaroszewski et al., 2005), SP3 (Zhou and Zhou, 2005), SALIGN (Marti-Renom et al., 2004), and PPSCAN (Eswar et al., 2005).

Sequence-structure threading methods

As the sequence identity drops below the threshold of the twilight zone, there is usually insufficient signal in the sequences or their profiles for the sequence-based methods discussed above to detect true relationships (Lindahl and Elofsson, 2000). Sequence-structure threading methods are most useful in this regime, as they can sometimes recognize common folds even in the absence of any statistically significant sequence similarity (Godzik, 2003). These methods achieve higher sensitivity by using structural information derived from the templates. The accuracy of a sequence-structure match is assessed by the score of a corresponding coarse model and not by sequence similarity, as in sequence-comparison methods (Godzik, 2003). The scoring scheme used to evaluate the accuracy is either based on residue substitution tables dependent on structural features such as solvent exposure, secondary structure type, and hydrogen-bonding properties (Shi et al., 2001; Karchin et al., 2003; McGuffin and Jones, 2003; Zhou and Zhou, 2005), or on statistical potentials for residue interactions implied by the alignment (Sippl, 1990; Bowie et al., 1991; Sippl, 1995; Skolnick and Kihara, 2001; Xu et al., 2003). The use of structural data does not have to be restricted to the structure side of the aligned sequence-structure pair. For example, SAM-T02 makes use of the predicted local structure for the target sequence to enhance homolog detection and alignment accuracy (Karplus et al., 2003). Commonly used threading programs are GenTHREADER (Jones, 1999; McGuffin and Jones, 2003), 3D-PSSM (Kelley et al., 2000), FUGUE (Shi et al., 2001), SP3 (Zhou and

Zhou, 2005), and SAM-T02 multi-track HMM (Karchin et al., 2003; Karplus et al., 2003).

Iterative sequence-structure alignment and model building.

Yet another strategy is to optimize the alignment by iterating over the process of calculating alignments, building models, and evaluating models. Such a protocol can sample alignments that are not statistically significant and identify the alignment that yields the best model. Although this procedure can be time consuming, it can significantly improve the accuracy of the resulting comparative models in difficult cases (John and Sali, 2003).

Importance of an accurate alignment

Regardless of the method used, searching in the twilight and midnight zones of the sequence-structure relationship often results in false negatives, false positives, or alignments that contain an increasingly large number of gaps and alignment errors. Improving the performance and accuracy of methods in this regime remains one of the main tasks of comparative modeling today (Moult, 2005). It is imperative to calculate an accurate alignment between the target-template pair, as comparative modeling can almost never recover from an alignment error (Sanchez and Sali, 1997a).

Template selection

After a list of all related protein structures and their alignments with the target sequence have been obtained, template structures are prioritized depending on the purpose of the comparative model. Template structures may be chosen based purely on the target-template sequence identity, or on a combination of several other criteria, such as experimental accuracy of the structures (resolution of X-ray structures, number of restraints per residue for NMR structures), conservation of active-site residues, holo-structures that have bound ligands of interest, and prior biological information that pertains to the solvent, pH, and quaternary contacts. It is not necessary to select only one template. In fact, the use of several templates approximately equidistant from the target sequence generally increases the model accuracy (Srinivasan and Blundell, 1993; Sanchez and Sali, 1997b).

Model building

Modeling by assembly of rigid bodies

The first and still widely used approach in comparative modeling is to assemble a model from a small number of rigid bodies obtained from the aligned protein structures (Browne et al., 1969; Greer, 1981; Blundell et al., 1987).

The approach is based on the natural dissection of the protein structures into conserved core regions, variable loops that connect them, and side chains that decorate the backbone. For example, the following semiautomated procedure is implemented in the computer program COMPOSER (Sutcliffe et al., 1987a). First, the template structures are selected and superposed. Second, the “framework” is calculated by averaging the coordinates of the C α atoms of structurally conserved regions in the template structures. Third, the main-chain atoms of each core region in the target model are obtained by superposing the core segment, from the template whose sequence is closest to the target, on the framework. Fourth, the loops are generated by scanning a database of all known protein structures to identify the structurally variable regions that fit the anchor core regions and have a compatible sequence (Topham et al., 1993). Fifth, the side chains are modeled based on their intrinsic conformational preferences and on the conformation of the equivalent side chains in the template structures (Sutcliffe et al., 1987b). Finally, the stereochemistry of the model is improved either by a restrained energy minimization or a molecular dynamics refinement. The accuracy of a model can be somewhat increased when more than one template structure is used to construct the framework and when the templates are averaged into the framework using weights corresponding to their sequence similarities to the target sequence (Srinivasan and Blundell, 1993). Possible future improvements of modeling by rigid-body assembly include incorporation of rigid body shifts, such as the relative shifts in the packing of a helices and β -sheets (Nagarajaram et al., 1999). Two other programs that implement this method are 3D-JIGSAW (Bates et al., 2001) and SWISS-MODEL (Schwede et al., 2003).

Modeling by segment matching or coordinate reconstruction

The basis of modeling by coordinate reconstruction is the finding that most hexapeptide segments of protein structure can be clustered into only 100 structurally different classes (Jones and Thirup, 1986; Claessens et al., 1989; Unger et al., 1989; Levitt, 1992; Bystroff and Baker, 1998). Thus, comparative models can be constructed by using a subset of atomic positions from template structures as guiding positions to identify and assemble short, all-atom segments that fit these guiding positions. The guiding positions usually correspond to the C α atoms of the

segments that are conserved in the alignment between the template structure and the target sequence. The all-atom segments that fit the guiding positions can be obtained either by scanning all known protein structures, including those that are not related to the sequence being modeled (Claessens et al., 1989; Holm and Sander, 1991), or by a conformational search restrained by an energy function (Brucoleri and Karplus, 1987; van Gelder et al., 1994). This method can construct both main-chain and side-chain atoms, and can also model unaligned regions (gaps). It is implemented in the program SegMod (Levitt, 1992). Even some side-chain modeling methods (Chinea et al., 1995) and the class of loop-construction methods based on finding suitable fragments in the database of known structures (Jones and Thirup, 1986) can be seen as segment-matching or coordinate-reconstruction methods.

Modeling by satisfaction of spatial restraints

The methods in this class begin by generating many constraints or restraints on the structure of the target sequence, using its alignment to related protein structures as a guide. The procedure is conceptually similar to that used in determination of protein structures from NMR-derived restraints. The restraints are generally obtained by assuming that the corresponding distances between aligned residues in the template and the target structures are similar. These homology-derived restraints are usually supplemented by stereochemical restraints on bond lengths, bond angles, dihedral angles, and nonbonded atom-atom contacts that are obtained from a molecular mechanics force field. The model is then derived by minimizing the violations of all the restraints. This optimization can be achieved either by distance geometry or real-space optimization. For example, an elegant distance geometry approach constructs all-atom models from lower and upper bounds on distances and dihedral angles (Havel and Snow, 1991).

Comparative protein structure modeling by MODELLER. MODELLER, the authors' own program for comparative modeling, belongs to this group of methods (Sali and Blundell, 1993; Sali and Overington, 1994; Fiser et al., 2000; Fiser et al., 2002). MODELLER implements comparative protein structure modeling by satisfaction of spatial restraints. The program was designed to use as many different types of information about the target sequence as possible.

Homology-derived restraints. In the first step of model building, distance and dihedral angle restraints on the target sequence are derived from its alignment with template 3-D structures. The form of these restraints was obtained from a statistical analysis of the relationships between similar protein structures. The analysis relied on a database of 105 family alignments that included 416 proteins of known 3-D structure (Sali and Overington, 1994). By scanning the database of alignments, tables quantifying various correlations were obtained, such as the correlations between two equivalent C α -C α distances, or between equivalent main-chain dihedral angles from two related proteins (Sali and Blundell, 1993). These relationships are expressed as conditional probability density functions (pdf's), and can be used directly as spatial restraints. For example, probabilities for different values of the main-chain dihedral angles are calculated from the type of residue considered, from main-chain conformation of an equivalent residue, and from sequence similarity between the two proteins. Another example is the pdf for a certain C α -C α distance given equivalent distances in two related protein structures. An important feature of the method is that the form of spatial restraints was obtained empirically, from a database of protein structure alignments.

Stereochemical restraints. In the second step, the spatial restraints and the CHARMM22 force-field terms enforcing proper stereochemistry (MacKerell et al., 1998) are combined into an objective function. The general form of the objective function is similar to that in molecular dynamics programs, such as CHARMM22 (MacKerell et al., 1998). The objective function depends on the Cartesian coordinates of $\sim 10,000$ atoms (3-D points) that form the modeled molecules. For a 10,000-atom system, there can be on the order of 200,000 restraints. The functional form of each term is simple; it includes a quadratic function, harmonic lower and upper bounds, cosine, a weighted sum of a few Gaussian functions, Coulomb law, Lennard-Jones potential, and cubic splines. The geometric features presently include a distance, an angle, a dihedral angle, a pair of dihedral angles between two, three, four, and eight atoms, respectively, the shortest distance in the set of distances, solvent accessibility, and atom density that is expressed as the number of atoms around the central atom. Some restraints can be used to restrain pseudo-atoms, e.g., the gravity center of several atoms.

Optimization of the objective function. Finally, the model is obtained by optimizing the objective function in Cartesian space. The optimization is carried out by the use of the variable target function method (Braun and Go, 1985), employing methods of conjugate gradients and molecular dynamics with simulated annealing (Clare et al., 1986). Several slightly different models can be calculated by varying the initial structure, and the variability among these models can be used to estimate the lower bound on the errors in the corresponding regions of the fold.

Restraints derived from experimental data. Because the modeling by satisfaction of spatial restraints can use many different types of information about the target sequence, it is perhaps the most promising of all comparative modeling techniques. One of the strengths of modeling by satisfaction of spatial restraints is that restraints derived from a number of different sources can easily be added to the homology-derived restraints. For example, restraints could be provided by rules for secondary-structure packing (Cohen et al., 1989), analyses of hydrophobicity (Aszodi and Taylor, 1994) and correlated mutations (Taylor et al., 1994), empirical potentials of mean force (Sippl, 1990), nuclear magnetic resonance (NMR) experiments (Sutcliffe et al., 1992), cross-linking experiments, fluorescence spectroscopy, image reconstruction in electron microscopy, site-directed mutagenesis (Boissel et al., 1993), and intuition, among other sources. Especially in difficult cases, a comparative model could be improved by making it consistent with available experimental data and/or with more general knowledge about protein structure.

Relative accuracy, flexibility, and automation. Accuracies of the various model-building methods are relatively similar when used optimally (Marti-Renom et al., 2002). Other factors such as template selection and alignment accuracy usually have a larger impact on the model accuracy, especially for models based on low sequence identity to the templates. However, it is important that a modeling method allow a degree of flexibility and automation to obtain better models more easily and rapidly. For example, a method should allow for an easy recalculation of a model when a change is made in the alignment. It should also be straightforward enough to calculate models based on several templates, and should provide tools for incorporation of prior knowledge about the target (e.g., cross-linking

restraints, predicted secondary structure) and allow ab initio modeling of insertions (e.g., loops), which can be crucial for annotation of function.

Loop modeling

Loop modeling is an especially important aspect of comparative modeling in the range from 30% to 50% sequence identity. In this range of overall similarity, loops among the homologs vary while the core regions are still relatively conserved and aligned accurately. Loops often play an important role in defining the functional specificity of a given protein, forming the active and binding sites. Loop modeling can be seen as a mini protein folding problem, because the correct conformation of a given segment of a polypeptide chain has to be calculated mainly from the sequence of the segment itself. However, loops are generally too short to provide sufficient information about their local fold. Even identical decapeptides in different proteins do not always have the same conformation (Kabsch and Sander, 1984; Mezei, 1998). Some additional restraints are provided by the core anchor regions that span the loop and by the structure of the rest of the protein that cradles the loop. Although many loop-modeling methods have been described, it is still challenging to correctly and confidently model loops longer than ~8 to 10 residues (Fiser et al., 2000; Jacobson et al., 2004).

There are two main classes of loop-modeling methods: (i) database search approaches that scan a database of all known protein structures to find segments fitting the anchor core regions (Jones and Thirup, 1986; Chothia and Lesk, 1987); (ii) conformational search approaches that rely on optimizing a scoring function (Moult and James, 1986; Brucoleri and Karplus, 1987; Shenkin et al., 1987). There are also methods that combine these two approaches (van Vlijmen and Karplus, 1997; Deane and Blundell, 2001).

Loop modeling by database search. The database search approach to loop modeling is accurate and efficient when a database of specific loops is created to address the modeling of the same class of loops, such as β -hairpins (Sibanda et al., 1989), or loops on a specific fold, such as the hypervariable regions in the immunoglobulin fold (Chothia and Lesk, 1987; Chothia et al., 1989). There are attempts to classify loop conformations into more general categories, thus extending the applicability of the database search approach (Ring et al., 1992; Oliva et al., 1997;

Rufino et al., 1997; Fernandez-Fuentes et al., 2006). However, the database methods are limited because the number of possible conformations increases exponentially with the length of a loop. As a result, only loops up to 4 to 7 residues long have most of their conceivable conformations present in the database of known protein structures (Fidelis et al., 1994; Lessel and Schomburg, 1994). This limitation is made even worse by the requirement for an overlap of at least one residue between the database fragment and the anchor core regions, which means that modeling a 5-residue insertion requires at least a 7-residue fragment from the database (Claessens et al., 1989). Despite the rapid growth of the database of known structures, it does not seem possible to cover most of the conformations of a 9-residue segment in the foreseeable future. On the other hand, most of the insertions in a family of homologous proteins are shorter than 10 to 12 residues (Fiser et al., 2000).

Loop modeling by conformational search. To overcome the limitations of the database search methods, conformational search methods were developed (Moult and James, 1986; Brucoleri and Karplus, 1987). There are many such methods, exploiting different protein representations, objective functions, and optimization or enumeration algorithms. The search algorithms include the minimum perturbation method (Fine et al., 1986), molecular dynamics simulations (Brucoleri and Karplus, 1990; van Vlijmen and Karplus, 1997), genetic algorithms (Ring et al., 1993), Monte Carlo and simulated annealing (Higo et al., 1992; Collura et al., 1993; Abagyan and Totrov, 1994), multiple copy simultaneous search (Zheng et al., 1993), self-consistent field optimization (Koehl and Delarue, 1995), and enumeration based on graph theory (Samudrala and Moult, 1998). The accuracy of loop predictions can be further improved by clustering the sampled loop conformations and partially accounting for the entropic contribution to the free energy (Xiang et al., 2002). Another way to improve the accuracy of loop predictions is to consider the solvent effects. Improvements in implicit solvation models, such as the Generalized Born solvation model, motivated their use in loop modeling. The solvent contribution to the free energy can be added to the scoring function for optimization, or it can be used to rank the sampled loop conformations after they are generated with a scoring function that does not include the solvent terms (Fiser et al., 2000; Felts et al., 2002; de Bakker et al., 2003; DePristo et al., 2003).

Loop modeling in MODELLER. The loop-modeling module in MODELLER implements the optimization-based approach (Fiser et al., 2000; Fiser and Sali, 2003b). The main reasons for choosing this implementation are the generality and conceptual simplicity of scoring function minimization, as well as the limitations on the database approach that are imposed by a relatively small number of known protein structures (Fidelis et al., 1994). Loop prediction by optimization is applicable to simultaneous modeling of several loops and loops interacting with ligands, which is not straightforward with the database-search approaches. Loop optimization in MODELLER relies on conjugate gradients and molecular dynamics with simulated annealing. The pseudo energy function is a sum of many terms, including some terms from the CHARMM22 molecular mechanics force field (MacKerell et al., 1998) and spatial restraints based on distributions of distances (Sippl, 1990; Melo et al., 2002) and dihedral angles in known protein structures. The method was tested on a large number of loops of known structure, both in the native and near-native environments (Fiser et al., 2000).

Comparative model building by iterative alignment, model building, and model assessment

Comparative or homology protein structure modeling is severely limited by errors in the alignment of a modeled sequence with related proteins of known three-dimensional structure. To ameliorate this problem, one can use an iterative method that optimizes both the alignment and the model implied by it (Sanchez and Sali, 1997a; Miwa et al., 1999). This task can be achieved by a genetic algorithm protocol that starts with a set of initial alignments and then iterates through realignment, model building, and model assessment to optimize a model assessment score (John and Sali, 2003). During this iterative process: (1) new alignments are constructed by the application of a number of genetic algorithm operators, such as alignment mutations and crossovers; (2) comparative models corresponding to these alignments are built by satisfaction of spatial restraints, as implemented in the program MODELLER; and (3) the models are assessed by a composite score, partly depending on an atomic statistical potential (Melo et al., 2002). When testing the procedure on a very difficult set of 19 modeling targets sharing only 4% to 27% sequence identity with their template structures,

the average final alignment accuracy increased from 37% to 45% relative to the initial alignment (the alignment accuracy was measured as the percentage of positions in the tested alignment that were identical to the reference structure-based alignment). Correspondingly, the average model accuracy increased from 43% to 54% (the model accuracy was measured as the percentage of the C α atoms of the model that were within 5 Å of the corresponding C α atoms in the superimposed native structure).

Errors in comparative models

As the similarity between the target and the templates decreases, the errors in the model increase. Errors in comparative models can be

divided into five categories (Sanchez and Sali, 1997a,b; Fig. 5.6.12), as follows:

Errors in side-chain packing (Fig. 5.6.12A). As the sequences diverge, the packing of side chains in the protein core changes. Sometimes even the conformation of identical side chains is not conserved, a pitfall for many comparative modeling methods. Side-chain errors are critical if they occur in regions that are involved in protein function, such as active sites and ligand-binding sites.

Distortions and shifts in correctly aligned regions (Fig. 5.6.12B). As a consequence of sequence divergence, the main-chain conformation changes, even if the overall fold remains the same. Therefore, it is possible that in some correctly aligned segments of a model

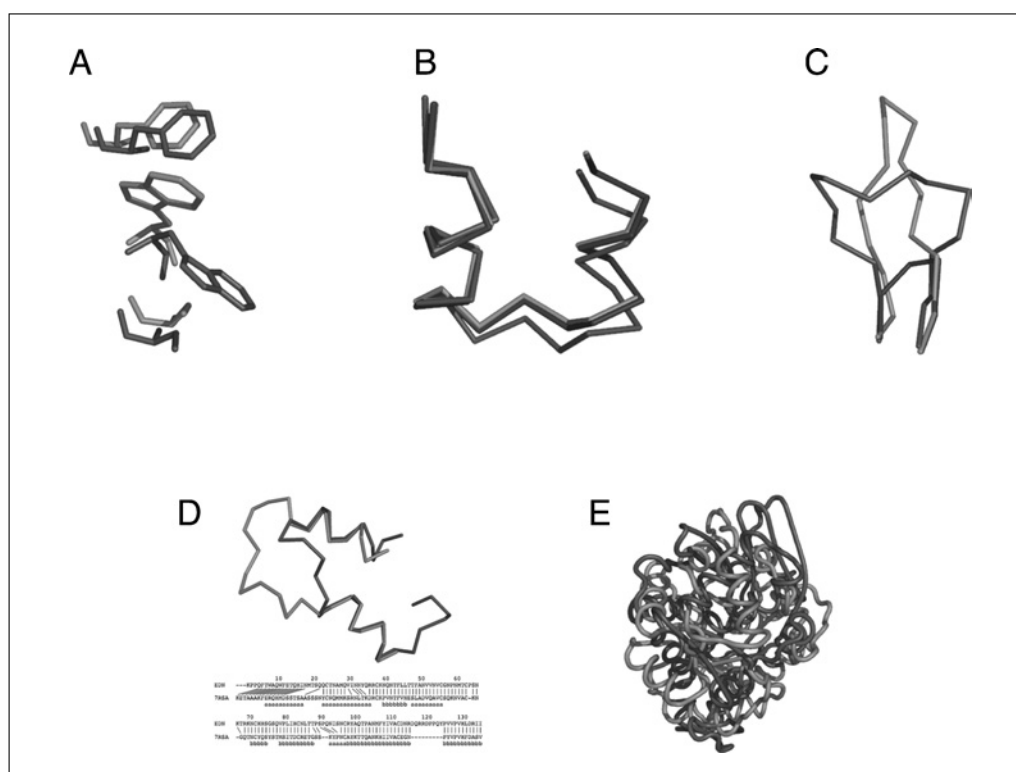


Figure 5.6.12 Typical errors in comparative modeling. **(A)** Errors in side chain packing. The Trp 109 residue in the crystal structure of mouse cellular retinoic acid binding protein I (red) is compared with its model (green). **(B)** Distortions and shifts in correctly aligned regions. A region in the crystal structure of mouse cellular retinoic acid binding protein I (red) is compared with its model (green) and with the template fatty acid binding protein (blue). **(C)** Errors in regions without a template. The C α trace of the 112–117 loop is shown for the X-ray structure of human eosinophil neurotoxin (red), its model (green), and the template ribonuclease A structure (residues 111–117; blue). **(D)** Errors due to misalignments. The N-terminal region in the crystal structure of human eosinophil neurotoxin (red) is compared with its model (green). The corresponding region of the alignment with the template ribonuclease A is shown. The red lines show correct equivalences, that is, residues whose C α atoms are within 5 Å of each other in the optimal least-squares superposition of the two X-ray structures. The “a” characters in the bottom line indicate helical residues and “b” characters, the residues in sheets. **(E)** Errors due to an incorrect template. The X-ray structure of α -trichosanthin (red) is compared with its model (green) that was calculated using indole-3-glycerophosphate synthase as the template. For the color version of this figure go to <http://www.currentprotocols.com>.

the template is locally different ($>3 \text{ \AA}$) from the target, resulting in errors in that region. The structural differences are sometimes not due to differences in sequence, but are a consequence of artifacts in structure determination or structure determination in different environments (e.g., packing of subunits in a crystal). The simultaneous use of several templates can minimize this kind of error (Srinivasan and Blundell, 1993; Sanchez and Sali, 1997a,b).

Errors in regions without a template (Fig. 5.6.12C). Segments of the target sequence that have no equivalent region in the template structure (i.e., insertions or loops) are the most difficult regions to model. If the insertion is relatively short, <9 residues long, some methods can correctly predict the conformation of the backbone (van Vlijmen and Karplus, 1997; Fiser et al., 2000; Jacobson et al., 2004). Conditions for successful prediction are the correct alignment and an accurately modeled environment surrounding the insertion.

Errors due to misalignments (Fig. 5.6.12D). The largest single source of errors in comparative modeling is misalignments, especially when the target-template sequence identity decreases below 30%. However, alignment errors can be minimized in two ways. First, it is usually possible to use a large number of sequences to construct a multiple alignment, even if most of these sequences do not have known structures. Multiple alignments are generally more reliable than pairwise alignments (Barton and Sternberg, 1987; Taylor et al., 1994). The second way of improving the alignment is to iteratively modify those regions in the alignment that correspond to predicted errors in the model (Sanchez and Sali, 1997a,b; John and Sali, 2003).

Incorrect templates (Fig. 5.6.12E). This is a potential problem when distantly related proteins are used as templates (i.e., $<25\%$ sequence identity). Distinguishing between a model based on an incorrect template and a model based on an incorrect alignment with a correct template is difficult. In both cases, the evaluation methods will predict an unreliable model. The conservation of the key functional or structural residues in the target sequence increases the confidence in a given fold assignment.

Predicting the model accuracy

The accuracy of the predicted model determines the information that can be extracted from it. Thus, estimating the accuracy of a model in the absence of the known structure is essential for interpreting it.

Initial assessment of the fold. As discussed earlier, a model calculated using a template structure that shares more than 30% sequence identity is indicative of an overall accurate structure. However, when the sequence identity is lower, the first aspect of model evaluation is to confirm whether or not a correct template was used for modeling. It is often the case, when operating in this regime, that the fold-assignment step produces only false positives. A further complication is that at such low similarities the alignment generally contains many errors, making it difficult to distinguish between an incorrect template on one hand and an incorrect alignment with a correct template on the other hand. There are several methods that use 3-D profiles and statistical potentials (Sippl, 1990; Luthy et al., 1992; Melo et al., 2002) to assess the compatibility between the sequence and modeled structure by evaluating the environment of each residue in a model with respect to the expected environment as found in native high-resolution experimental structures. These methods can be used to assess whether or not the correct template was used for the modeling. They include VERIFY3D (Luthy et al., 1992), PROSAIL (Sippl, 1993), HARMONY (Topham et al., 1994), ANOLEA (Melo and Feytmans, 1998), and DFIRE (Zhou and Zhou, 2002).

Even when the model is based on alignments that have $>30\%$ sequence identity, other factors, including the environment, can strongly influence the accuracy of a model. For instance, some calcium-binding proteins undergo large conformational changes when bound to calcium. If a calcium-free template is used to model the calcium-bound state of the target, it is likely that the model will be incorrect irrespective of the target-template similarity or accuracy of the template structure (Pawlowski et al., 1996).

Evaluations of self-consistency. The model should also be subjected to evaluations of self-consistency to ensure that it satisfies the restraints used to calculate it. Additionally, the stereochemistry of the model (e.g., bond-lengths, bond-angles, backbone torsion angles, and nonbonded contacts) may be evaluated using programs such as PROCHECK (Laskowski et al., 1993) and WHATCHECK (Hooft et al., 1996). Although errors in stereochemistry are rare and less informative than errors detected by statistical potentials, a cluster of stereochemical errors may indicate that there are larger errors (e.g., alignment errors) in that region.

Applications

Comparative modeling is often an efficient way to obtain useful information about the protein of interest. For example, comparative models can be helpful in designing mutants to test hypotheses about the protein's function (Wu et al., 1999; Vernal et al., 2002); in identifying active and binding sites (Sheng et al., 1996); in searching for, designing, and improving ligand binding strength for a given binding site (Ring et al., 1993; Li et al., 1996; Selzer et al., 1997; Enyedy et al., 2001; Que et al., 2002); modeling substrate specificity (Xu et al., 1996); in predicting antigenic epitopes (Sali and Blundell, 1993); in simulating protein-protein docking (Vakser, 1995); in inferring function from calculated electrostatic potential around the protein (Matsumoto et al., 1995); in facilitating molecular replacement in X-ray structure determination (Howell et al., 1992); in refining models based on NMR constraints (Modi et al., 1996); in testing and improving a sequence-structure alignment (Wolf et al., 1998); in annotating single nucleotide polymorphisms (Mirkovic et al., 2004; Karchin et al., 2005); in structural characterization of large complexes by docking to low-resolution cryo-electron density maps (Spahn et al., 2001; Gao et al., 2003); and in rationalizing known experimental observations.

Fortunately, a 3-D model does not have to be absolutely perfect to be helpful in biology, as demonstrated by the applications listed above. The type of a question that can be addressed with a particular model does depend on its accuracy (Fig. 5.6.13).

At the low end of the accuracy spectrum, there are models that are based on less than 25% sequence identity and that sometimes have less than 50% of their C α atoms within 3.5 Å of their correct positions. However, such models still have the correct fold, and even knowing only the fold of a protein may sometimes be sufficient to predict its approximate biochemical function. Models in this low range of accuracy, combined with model evaluation, can be used for confirming or rejecting a match between remotely related proteins (Sanchez and Sali, 1997a; 1998).

In the middle of the accuracy spectrum are the models based on approximately 35% sequence identity, corresponding to 85% of the C α atoms modeled within 3.5 Å of their correct positions. Fortunately, the active and binding sites are frequently more conserved than the rest of the fold, and are thus modeled more accurately (Sanchez and Sali, 1998). In general, medium-resolution models frequently allow a

refinement of the functional prediction based on sequence alone, because ligand binding is most directly determined by the structure of the binding site rather than its sequence. It is frequently possible to correctly predict important features of the target protein that do not occur in the template structure. For example, the location of a binding site can be predicted from clusters of charged residues (Matsumoto et al., 1995), and the size of a ligand may be predicted from the volume of the binding-site cleft (Xu et al., 1996). Medium-resolution models can also be used to construct site-directed mutants with altered or destroyed binding capacity, which in turn could test hypotheses about the sequence-structure-function relationships. Other problems that can be addressed with medium-resolution comparative models include designing proteins that have compact structures, without long tails, loops, and exposed hydrophobic residues, for better crystallization, or designing proteins with added disulfide bonds for extra stability.

The high end of the accuracy spectrum corresponds to models based on 50% sequence identity or more. The average accuracy of these models approaches that of low-resolution X-ray structures (3 Å resolution) or medium-resolution NMR structures (10 distance restraints per residue; Sanchez and Sali, 1997b). The alignments on which these models are based generally contain almost no errors. Models with such high accuracy have been shown to be useful even for refining crystallographic structures by the method of molecular replacement (Howell et al., 1992; Baker and Sali, 2001; Jones, 2001; Claude et al., 2004; Schwarzenbacher et al., 2004).

Conclusion

Over the past few years, there has been a gradual increase in both the accuracy of comparative models and the fraction of protein sequences that can be modeled with useful accuracy (Marti-Renom et al., 2000; Baker and Sali, 2001; Pieper et al., 2006). The magnitude of errors in fold assignment, alignment, and the modeling of side-chains and loops have decreased considerably. These improvements are a consequence both of better techniques and a larger number of known protein sequences and structures. Nevertheless, all the errors remain significant and demand future methodological improvements. In addition, there is a great need for more accurate modeling of distortions and rigid-body shifts, as well as detection of errors in a given protein structure model. Error detection is useful

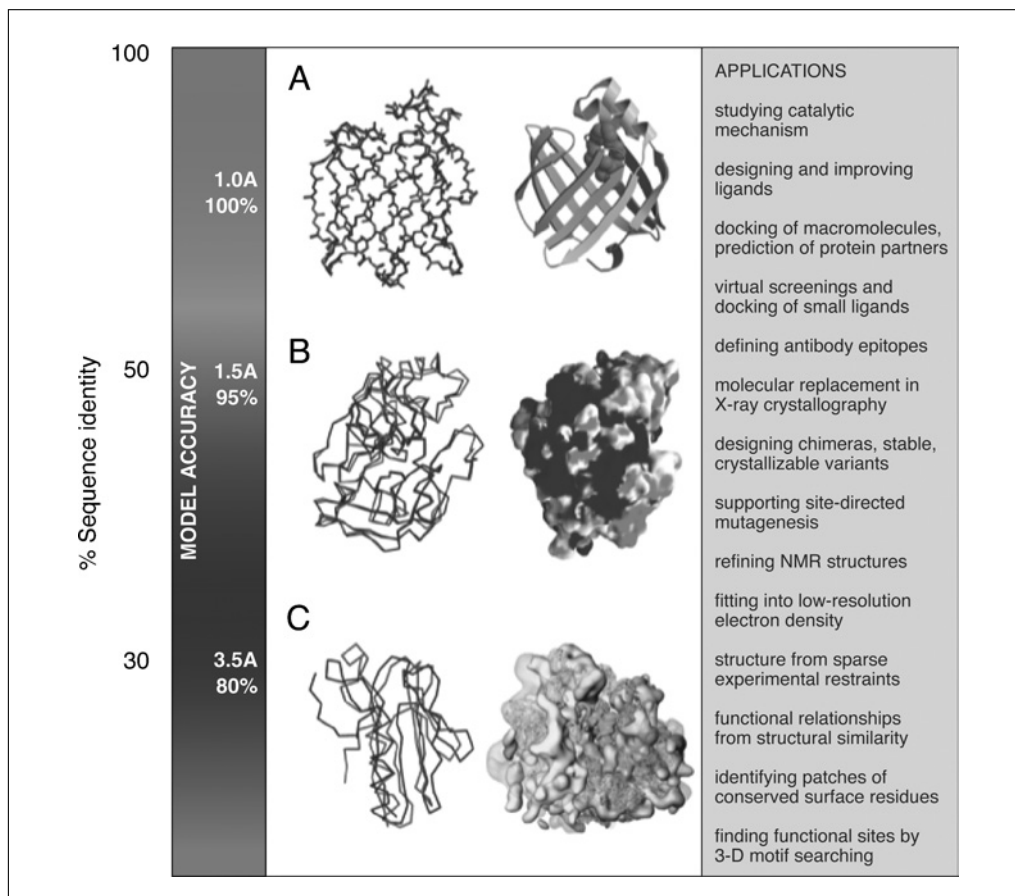


Figure 5.6.13 ptAccuracy and application of protein structure models. The vertical axis indicates the different ranges of applicability of comparative protein structure modeling, the corresponding accuracy of protein structure models, and their sample applications. **(A)** The docosaheptaenoic fatty acid ligand (violet) was docked into a high accuracy comparative model of brain lipid-binding protein (right), modeled based on its 62% sequence identity to the crystallographic structure of adipocyte lipid-binding protein (PDB code *1adl*). A number of fatty acids were ranked for their affinity to brain lipid-binding protein consistently with site-directed mutagenesis and affinity chromatography experiments (Xu et al., 1996), even though the ligand specificity profile of this protein is different from that of the template structure. Typical overall accuracy of a comparative model in this range of sequence similarity is indicated by a comparison of a model for adipocyte fatty acid binding protein with its actual structure (left). **(B)** A putative proteoglycan binding patch was identified on a medium-accuracy comparative model of mouse mast cell protease 7 (right), modeled based on its 39% sequence identity to the crystallographic structure of bovine pancreatic trypsin (*2ptn*) that does not bind proteoglycans. The prediction was confirmed by site-directed mutagenesis and heparin-affinity chromatography experiments (Matsumoto et al., 1995). Typical accuracy of a comparative model in this range of sequence similarity is indicated by a comparison of a trypsin model with the actual structure. **(C)** A molecular model of the whole yeast ribosome (right) was calculated by fitting atomic rRNA and protein models into the electron density of the 80S ribosomal particle, obtained by electron microscopy at 15 Å resolution (Spahn et al., 2001). Most of the models for 40 out of the 75 ribosomal proteins were based on template structures that were approximately 30% sequentially identical. Typical accuracy of a comparative model in this range of sequence similarity is indicated by a comparison of a model for a domain in L2 protein from *B. Stearothermophilus* with the actual structure (*1rl2*). For the color version of this figure go to <http://www.currentprotocols.com>.

both for refinement and interpretation of the models.

Acknowledgments

The authors wish to express gratitude to all members of their research group. This review is partially based on the authors' previous reviews (Marti-Renom et al., 2000; Eswar et al., 2003; Fiser and Sali, 2003a). They wish acknowledge funding from Sandler Family Supporting Foundation, NIH R01 GM54762, P01 GM71790, P01 A135707, and U54 GM62529, as well as hardware gifts from IBM and Intel.

Literature Cited

- Abagyan, R. and Totrov, M. 1994. Biased probability Monte Carlo conformational searches and electrostatic calculations for peptides and proteins. *J. Mol. Biol.* 235:983-1002.
- Alexandrov, N.N., Nussinov, R., and Zimmer, R.M. 1996. Fast protein fold recognition via sequence to structure alignment and contact capacity potentials. *Pac. Symp. Biocomput.* 1996:53-72.
- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25:3389-3402.
- Andreeva, A., Howorth, D., Brenner, S.E., Hubbard, T.J., Chothia, C., and Murzin, A.G. 2004. SCOP database in 2004: Refinements integrate structure and sequence family data. *Nucl. Acids Res.* 32:D226-D229.
- Aszodi, A. and Taylor, W.R. 1994. Secondary structure formation in model polypeptide chains. *Protein Eng.* 7:633-644.
- Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., and Yeh, L.S. 2005. The Universal Protein Resource (UniProt). *Nucl. Acids Res.* 33:D154-D159.
- Baker, D. and Sali, A. 2001. Protein structure prediction and structural genomics. *Science* 294:93-96.
- Barton, G.J. and Sternberg, M.J. 1987. A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons. *J. Mol. Biol.* 198:327-337.
- Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E.L., Studholme, D.J., Yeats, C., and Eddy, S.R. 2004. The Pfam protein families database. *Nucl. Acids Res.* 32:D138-D141.
- Bates, P.A., Kelley, L.A., MacCallum, R.M., and Sternberg, M.J. 2001. Enhancement of protein modeling by human intervention in applying the automatic programs 3D-JIGSAW and 3D-PSSM. *Proteins* 5:39-46.
- Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., and Wheeler, D.L. 2005. GenBank. *Nucl. Acids Res.* 33:D34-D38.
- Blundell, T.L., Sibanda, B.L., Sternberg, M.J., and Thornton, J.M. 1987. Knowledge-based prediction of protein structures and the design of novel molecules. *Nature* 326:347-352.
- Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.C., Estreicher, A., Gasteiger, E., Martin, M.J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., and Schneider, M. 2003. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucl. Acids Res.* 31:365-370.
- Boissel, J.P., Lee, W.R., Presnell, S.R., Cohen, F.E., and Bunn, H.F. 1993. Erythropoietin structure-function relationships: Mutant proteins that test a model of tertiary structure. *J. Biol. Chem.* 268:15983-15993.
- Bowie, J.U., Luthy, R., and Eisenberg, D. 1991. A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253:164-170.
- Braun, W. and Go, N. 1985. Calculation of protein conformations by proton-proton distance constraints: A new efficient algorithm. *J. Mol. Biol.* 186:611-626.
- Brenner, S.E., Chothia, C., and Hubbard, T.J. 1998. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. U.S.A.* 95:6073-6078.
- Browne, W.J., North, A.C., Phillips, D.C., Brew, K., Vanaman, T.C., and Hill, R.L. 1969. A possible three-dimensional structure of bovine alpha-lactalbumin based on that of hen's egg-white lysozyme. *J. Mol. Biol.* 42:65-86.
- Bruccoleri, R.E. and Karplus, M. 1987. Prediction of the folding of short polypeptide segments by uniform conformational sampling. *Biopolymers* 26:137-168.
- Bruccoleri, R.E. and Karplus, M. 1990. Conformational sampling using high-temperature molecular dynamics. *Biopolymers* 29:1847-1862.
- Bujnicki, J.M., Elofsson, A., Fischer, D., and Rychlewski, L. 2001. LiveBench-1: Continuous benchmarking of protein structure prediction servers. *Protein Sci.* 10:352-361.
- Bystroff, C. and Baker, D. 1998. Prediction of local structure in proteins using a library of sequence-structure motifs. *J. Mol. Biol.* 281:565-577.
- Canutescu, A.A., Shelenkov, A.A., and Dunbrack, R.L. Jr. 2003. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Sci.* 12:2001-2014.
- Chinea, G., Padron, G., Hooft, R.W., Sander, C., and Vriend, G. 1995. The use of position-specific rotamers in model building by homology. *Proteins* 23:415-421.
- Chothia, C. and Lesk, A.M. 1987. Canonical structures for the hypervariable regions of immunoglobulins. *J. Mol. Biol.* 196:901-917.

- Chothia, C., Lesk, A.M., Tramontano, A., Levitt, M., Smith-Gill, S.J., Air, G., Sheriff, S., Padlan, E.A., Davies, D., Tulip, W.R., Colman, P.M., Spinelli, S., Alzari, P.M., and Poljak, J. 1989. Conformations of immunoglobulin hypervariable regions. *Nature* 342:877-883.
- Claessens, M., Van Cutsem, E., Lasters, I., and Wodak, S. 1989. Modelling the polypeptide backbone with 'spare parts' from known protein structures. *Protein Eng.* 2:335-345.
- Claude, J.B., Suhre, K., Notredame, C., Claverie, J.M., and Abergel, C. 2004. CaspR: A web server for automated molecular replacement using homology modelling. *Nucl. Acids Res.* 32:W606-W609.
- Clore, G.M., Brunger, A.T., Karplus, M., and Gronenborn, A.M. 1986. Application of molecular dynamics with interproton distance restraints to three-dimensional protein structure determination: A model study of crambin. *J. Mol. Biol.* 191:523-551.
- Cohen, F.E., Gregoret, L., Presnell, S.R., and Kuntz, I.D. 1989. Protein structure predictions: New theoretical approaches. *Prog. Clin. Biol. Res.* 289:75-85.
- Collura, V., Higo, J., and Garnier, J. 1993. Modeling of protein loops by simulated annealing. *Protein Sci.* 2:1502-1510.
- Colovos, C. and Yeates, T.O. 1993. Verification of protein structures: Patterns of nonbonded atomic interactions. *Protein Sci.* 2:1511-1519.
- Corpet, F. 1988. Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.* 16:10881-10890.
- Deane, C.M. and Blundell, T.L. 2001. CODA: A combined algorithm for predicting the structurally variable regions of protein models. *Protein Sci.* 10:599-612.
- de Bakker, P.I., DePristo, M.A., Burke, D.F., and Blundell, T.L. 2003. Ab initio construction of polypeptide fragments: Accuracy of loop decoy discrimination by an all-atom statistical potential and the AMBER force field with the Generalized Born solvation model. *Proteins* 51:21-40.
- DePristo, M.A., de Bakker, P.I., Lovell, S.C., and Blundell, T.L. 2003. Ab initio construction of polypeptide fragments: Efficient generation of accurate, representative ensembles. *Proteins* 51:41-55.
- Deshpande, N., Address, K.J., Bluhm, W.F., Merino-Ott, J.C., Townsend-Merino, W., Zhang, Q., Knezevich, C., Xie, L., Chen, L., Feng, Z., Green, R.K., Flippen-Anderson, J.L., Westbrook, J., Berman, H.M., and Bourne, P.E. 2005. The RCSB Protein Data Bank: A redesigned query system and relational database based on the mmCIF schema. *Nucl. Acids Res.* 33:D233-D237.
- Dietmann, S., Park, J., Notredame, C., Heger, A., Lappe, M., and Holm, L. 2001. A fully automatic evolutionary classification of protein folds: Dali Domain Dictionary version 3. *Nucl. Acids Res.* 29:55-57.
- Eddy, S.R. 1998. Profile hidden Markov models. *Bioinformatics* 14:755-763.
- Edgar, R.C. 2004. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucl. Acids Res.* 32:1792-1797.
- Edgar, R.C. and Sjolander, K. 2004. A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics* 20:1301-1308.
- Enyedy, I.J., Ling, Y., Nacro, K., Tomita, Y., Wu, X., Cao, Y., Guo, R., Li, B., Zhu, X., Huang, Y., Long, Y.Q., Roller, P.P., Yang, D., and Wang, S. 2001. Discovery of small-molecule inhibitors of Bcl-2 through structure-based computer screening. *J. Med. Chem.* 44:4313-4324.
- Eswar, N., John, B., Mirkovic, N., Fiser, A., Ilyin, V.A., Pieper, U., Stuart, A.C., Marti-Renom, M.A., Madhusudhan, M.S., Yerkovich, B., and Sali, A. 2003. Tools for comparative protein structure modeling and analysis. *Nucl. Acids Res.* 31:3375-3380.
- Eyrich, V.A., Marti-Renom, M.A., Przybylski, D., Madhusudhan, M.S., Fiser, A., Pazos, F., Valencia, A., Sali, A., and Rost, B. 2001. EVA: Continuous automatic evaluation of protein structure prediction servers. *Bioinformatics* 17:1242-1243.
- Felsenstein, J. 1989. PHYLIP—Phylogeny Inference Package (Version 3.2). *Cladistics* 5:164-166.
- Felts, A.K., Gallicchio, E., Wallqvist, A., and Levy, R.M. 2002. Distinguishing native conformations of proteins from decoys with an effective free energy estimator based on the OPLS all-atom force field and the surface generalized born solvent model. *Proteins* 48:404-422.
- Fernandez-Fuentes, N., Oliva, B., and Fiser, A. 2006. A supersecondary structure library and search algorithm for modeling loops in protein structures. *Nucl. Acids Res.* 34:2085-2097.
- Fidelis, K., Stern, P.S., Bacon, D., and Moulton, J. 1994. Comparison of systematic search and database methods for constructing segments of protein structure. *Protein Eng.* 7:953-960.
- Fine, R.M., Wang, H., Shenkin, P.S., Yarmush, D.L., and Levinthal, C. 1986. Predicting antibody hypervariable loop conformations. II: Minimization and molecular dynamics studies of MCPC603 from many randomly generated loop conformations. *Proteins* 1:342-362.
- Fischer, D. 2006. Servers for protein structure prediction. *Curr. Opin. Struct. Biol.* 16:178-182.
- Fischer, D., Elofsson, A., Rychlewski, L., Pazos, F., Valencia, A., Rost, B., Ortiz, A.R., and Dunbrack, R.L. Jr., 2001. CAFASP2: The second critical assessment of fully automated structure prediction methods. *Proteins* 5:171-183.
- Fiser, A. 2004. Protein structure modeling in the proteomics era. *Expert Rev. Proteomics* 1:97-110.
- Fiser, A. and Sali, A. 2003a. Modeller: Generation and refinement of homology-based protein structure models. *Methods Enzymol.* 374:461-491.

- Fiser, A. and Sali, A. 2003b. ModLoop: Automated modeling of loops in protein structures. *Bioinformatics* 19:2500-2501.
- Fiser, A., Do, R.K., and Sali, A. 2000. Modeling of loops in protein structures. *Protein Sci.* 9:1753-1773.
- Fiser, A., Feig, M., Brooks, C.L. 3rd, and Sali, A. 2002. Evolution and physics in comparative protein structure modeling. *Acc. Chem. Res.* 35:413-421.
- Gao, H., Sengupta, J., Valle, M., Korostelev, A., Eswar, N., Stagg, S.M., Van Roey, P., Agrawal, R.K., Harvey, S.C., Sali, A., Chapman, M.S., and Frank, J. 2003. Study of the structural dynamics of the E coli 70S ribosome using real-space refinement. *Cell* 113:789-801.
- Godzik, A. 2003. Fold recognition methods. *Methods Biochem. Anal.* 44:525-546.
- Gough, J., Karplus, K., Hughey, R., and Chothia, C. 2001. Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *J. Mol. Biol.* 313:903-919.
- Greer, J. 1981. Comparative model-building of the mammalian serine proteases. *J. Mol. Biol.* 153:1027-1042.
- Gribskov, M., McLachlan, A.D., and Eisenberg, D. 1987. Profile analysis: Detection of distantly related proteins. *Proc. Natl. Acad. Sci. U.S.A.* 84:4355-4358.
- Havel, T.F. and Snow, M.E. 1991. A new method for building protein conformations from sequence alignments with homologues of known structure. *J. Mol. Biol.* 217:1-7.
- Henikoff, J.G. and Henikoff, S. 1996. Using substitution probabilities to improve position-specific scoring matrices. *Comput. Appl. Biosci.* 12:135-143.
- Henikoff, J.G., Pietrokovski, S., McCallum, C.M., and Henikoff, S. 2000. Blocks-based methods for detecting protein homology. *Electrophoresis* 21:1700-1706.
- Henikoff, S. and Henikoff, J.G. 1994. Position-based sequence weights. *J. Mol. Biol.* 243:574-578.
- Higo, J., Collura, V., and Garnier, J. 1992. Development of an extended simulated annealing method: Application to the modeling of complementary determining regions of immunoglobulins. *Biopolymers* 32:33-43.
- Holm, L. and Sander, C. 1991. Database algorithm for generating protein backbone and side-chain co-ordinates from a C alpha trace application to model building and detection of co-ordinate errors. *J. Mol. Biol.* 218:183-194.
- Hooft, R.W., Vriend, G., Sander, C., and Abola, E.E. 1996. Errors in protein structures. *Nature* 381:272.
- Howell, P.L., Almo, S.C., Parsons, M.R., Hajdu, J., and Petsko, G.A. 1992. Structure determination of turkey egg-white lysozyme using Laue diffraction data. *Acta Crystallogr. B* 48:200-207.
- Jacobson, M.P., Pincus, D.L., Rapp, C.S., Day, T.J., Honig, B., Shaw, D.E., and Friesner, R.A. 2004. A hierarchical approach to all-atom protein loop prediction. *Proteins* 55:351-367.
- Jaroszewski, L., Rychlewski, L., Li, Z., Li, W., and Godzik, A. 2005. FFAS03: A server for profile-profile sequence alignments. *Nucl. Acids Res.* 33:W284-W288.
- John, B. and Sali, A. 2003. Comparative protein structure modeling by iterative alignment, model building and model assessment. *Nucl. Acids Res.* 31:3982-3992.
- Jones, D.T. 1999. GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.* 287:797-815.
- Jones, D.T. 2001. Evaluating the potential of using fold-recognition models for molecular replacement. *Acta Crystallogr. D Biol. Crystallogr.* 57:1428-1434.
- Jones, D.T., Taylor, W.R., and Thornton, J.M. 1992. A new approach to protein fold recognition. *Nature* 358:86-89.
- Jones, T.A. and Thirup, S. 1986. Using known substructures in protein model building and crystallography. *Embo J.* 5:819-822.
- Kabsch, W. and Sander, C. 1984. On the use of sequence homologies to predict protein structure: Identical pentapeptides can have completely different conformations. *Proc. Natl. Acad. Sci. U.S.A.* 81:1075-1078.
- Kahsay, R.Y., Wang, G., Dongre, N., Gao, G., and Dunbrack, R.L. Jr. 2002. CASA: A server for the critical assessment of protein sequence alignment accuracy. *Bioinformatics* 18:496-497.
- Karchin, R., Cline, M., Mandel-Gutfreund, Y., and Karplus, K. 2003. Hidden Markov models that use predicted local structure for fold recognition: Alphabets of backbone geometry. *Proteins* 51:504-514.
- Karchin, R., Diekhans, M., Kelly, L., Thomas, D.J., Pieper, U., Eswar, N., Haussler, D., and Sali, A. 2005. LS-SNP: Large-scale annotation of coding non-synonymous SNPs based on multiple information sources. *Bioinformatics* 21:2814-2820.
- Karplus, K., Barrett, C., and Hughey, R. 1998. Hidden Markov models for detecting remote protein homologies. *Bioinformatics* 14:846-856.
- Karplus, K., Karchin, R., Draper, J., Casper, J., Mandel-Gutfreund, Y., Diekhans, M., and Hughey, R. 2003. Combining local-structure, fold-recognition, and new fold methods for protein structure prediction. *Proteins* 53:491-496.
- Kelley, L.A., MacCallum, R.M., and Sternberg, M.J. 2000. Enhanced genome annotation using structural profiles in the program 3D-PSSM. *J. Mol. Biol.* 299:499-520.
- Koehl, P. and Delarue, M. 1995. A self consistent mean field approach to simultaneous gap closure and side-chain positioning in homology modelling. *Nat. Struct. Biol.* 2:163-170.

- Koh, I.-Y.Y., Eyrich, V.A., Marti-Renom, M.A., Przybylski, D., Madhusudhan, M.S., Narayanan, E., Grana, O., Pazos, F., Valencia, A., Sali, A., and Rost, B. 2003. EVA: Evaluation of protein structure prediction servers. *Nucl. Acids Res.* 31:3311-3315.
- Krogh, A., Brown, M., Mian, I.S., Sjolander, K., and Haussler, D. 1994. Hidden Markov models in computational biology. Applications to protein modeling. *J. Mol. Biol.* 235:1501-1531.
- Laskowski, R.A., MacArthur, M.W., Moss, D.S., and Thornton, J.M. 1993. PROCHECK: A program to check the stereochemical quality of protein structures. *J. Appl. Crystallogr.* 26:283-291.
- Laskowski, R.A., Rullmann, J.A., MacArthur, M.W., Kaptein, R., and Thornton, J.M. 1996. AQUA and PROCHECK-NMR: Programs for checking the quality of protein structures solved by NMR. *J. Biomol. NMR* 8:477-486.
- Laskowski, R.A., MacArthur, M.W., and Thornton, J.M. 1998. Validation of protein models derived from experiment. *Curr. Opin. Struct. Biol.* 8:631-639.
- Lessel, U. and Schomburg, D. 1994. Similarities between protein 3-D structures. *Protein Eng.* 7:1175-1187.
- Levitt, M. 1992. Accurate modeling of protein conformation by automatic segment matching. *J. Mol. Biol.* 226:507-533.
- Li, R., Chen, X., Gong, B., Selzer, P.M., Li, Z., Davidson, E., Kurzbach, G., Miller, R.E., Nuzum, E.O., McKerrow, J.H., Fletterick, R.J., Gillmor, S.A., Craik, C.S., Kuntz, I.D., Cohen, F.E., and Kenyon, G.L. 1996. Structure-based design of parasitic protease inhibitors. *Bioorg. Med. Chem.* 4:1421-1427.
- Lin, J., Qian, J., Greenbaum, D., Bertone, P., Das, R., Echols, N., Senes, A., Stenger, B., and Gerstein, M. 2002. GeneCensus: Genome comparisons in terms of metabolic pathway activity and protein family sharing. *Nucl. Acids Res.* 30:4574-4582.
- Lindahl, E. and Elofsson, A. 2000. Identification of related proteins on family, superfamily and fold level. *J. Mol. Biol.* 295:613-625.
- Luthy, R., Bowie, J.U., and Eisenberg, D. 1992. Assessment of protein models with three-dimensional profiles. *Nature* 356:83-85.
- MacKerell, A.D. Jr., Bashford, D., Bellott, M., Dunbrack, R.L. Jr., Evanseck, J.D., Field, M.J., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kucera, K., Lau, F.T.K., Mattos, C., Michnick, S., Ngo, T., Nguyen, D.T., Prodhom, B., Reiher, W.E. III, Roux, B., Schlenkrich, M., Smith, J.C., Stote, R., Straub, J., Watanabe, M., Wiorkiewicz-Kucera, J., Yin, D., and Karplus, M. 1998. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B* 102:3586-3616.
- Madhusudhan, M.S., Marti-Renom, M.A., Sanchez, R., and Sali, A. 2006. Variable gap penalty for protein sequence-structure alignment. *Protein Eng. Des. Sel.* 19:129-133.
- Mallick, P., Weiss, R., and Eisenberg, D. 2002. The directional atomic solvation energy: An atom-based potential for the assignment of protein sequences to known folds. *Proc. Natl. Acad. Sci. U.S.A.* 99:16041-16046.
- Marti-Renom, M.A., Stuart, A.C., Fiser, A., Sanchez, R., Melo, F., and Sali, A. 2000. Comparative protein structure modeling of genes and genomes. *Annu. Rev. Biophys. Biomol. Struct.* 29:291-325.
- Marti-Renom, M.A., Ilyin, V.A., and Sali, A. 2001. DBali: A database of protein structure alignments. *Bioinformatics* 17:746-747.
- Marti-Renom, M.A., Madhusudhan, M.S., Fiser, A., Rost, B., and Sali, A. 2002. Reliability of assessment of protein structure prediction methods. *Structure (Camb)* 10:435-440.
- Marti-Renom, M.A., Madhusudhan, M.S., and Sali, A. 2004. Alignment of protein sequences by their profiles. *Protein Sci.* 13:1071-1087.
- Matsumoto, R., Sali, A., Ghildyal, N., Karplus, M., and Stevens, R.L. 1995. Packaging of proteases and proteoglycans in the granules of mast cells and other hematopoietic cells. A cluster of histidines on mouse mast cell protease 7 regulates its binding to heparin serglycin proteoglycans. *J. Biol. Chem.* 270:19524-19531.
- McGuffin, L.J. and Jones, D.T. 2003. Improvement of the GenTHREADER method for genomic fold recognition. *Bioinformatics* 19:874-881.
- McGuffin, L.J., Bryson, K., and Jones, D.T. 2000. The PSIPRED protein structure prediction server. *Bioinformatics* 16:404-405.
- Melo, F. and Feytmans, E. 1998. Assessing protein structures with a non-local atomic interaction energy. *J. Mol. Biol.* 277:1141-1152.
- Melo, F., Sanchez, R., and Sali, A. 2002. Statistical potentials for fold assessment. *Protein Sci.* 11:430-448.
- Mezei, M. 1998. Chameleon sequences in the PDB. *Protein Eng.* 11:411-414.
- Mirkovic, N., Marti-Renom, M.A., Sali, A., and Monteiro, A.N.A. 2004. Structure-based assessment of missense mutations in human BRCA1: Implications for breast and ovarian cancer predisposition. *Cancer Res.* 64:3790-3797.
- Misura, K.M. and Baker, D. 2005. Progress and challenges in high-resolution refinement of protein structure models. *Proteins* 59:15-29.
- Misura, K.M., Chivian, D., Rohl, C.A., Kim, D.E., and Baker, D. 2006. Physically realistic homology models built with ROSETTA can be more accurate than their templates. *Proc. Natl. Acad. Sci. U.S.A.* 103:5361-5366.
- Miwa, J.M., Ibanez-Tallon, I., Crabtree, G.W., Sanchez, R., Sali, A., Role, L.W., and Heintz, N. 1999. lynx1, an endogenous toxin-like modulator of nicotinic acetylcholine receptors in the mammalian CNS. *Neuron* 23:105-114.
- Modi, S., Paine, M.J., Sutcliffe, M.J., Lian, L.Y., Primrose, W.U., Wolf, C.R., and Roberts, G.C. 1996. A model for human cytochrome P450 2D6 based on homology modeling and NMR studies

- of substrate binding. *Biochemistry* 35:4540-4550.
- Moult, J. 2005. A decade of CASP: Progress, bottlenecks and prognosis in protein structure prediction. *Curr. Opin. Struct. Biol.* 15:285-289.
- Moult, J. and James, M.N. 1986. An algorithm for determining the conformation of polypeptide segments in proteins by systematic search. *Proteins* 1:146-163.
- Moult, J., Fidelis, K., Zemla, A., and Hubbard, T. 2003. Critical assessment of methods of protein structure prediction (CASP)-round V. *Proteins* 53:334-339.
- Moult, J., Fidelis, K., Rost, B., Hubbard, T., and Tramontano, A. 2005. Critical assessment of methods of protein structure prediction (CASP)-round 6. *Proteins* 61:3-7.
- Nagarajaram, H.A., Reddy, B.V., and Blundell, T.L. 1999. Analysis and prediction of inter-strand packing distances between beta-sheets of globular proteins. *Protein Eng.* 12:1055-1062.
- Needleman, S.B. and Wunsch, C.D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48:443-453.
- Notredame, C., Higgins, D.G., and Heringa, J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* 302:205-217.
- Ohlson, T., Wallner, B., and Elofsson, A. 2004. Profile-profile methods provide improved fold-recognition: A study of different profile-profile alignment methods. *Proteins* 57:188-197.
- Oldfield, T.J. 1992. SQUID: A program for the analysis and display of data from crystallography and molecular dynamics. *J. Mol. Graph.* 10:247-252.
- Oliva, B., Bates, P.A., Querol, E., Aviles, F.X., and Sternberg, M.J. 1997. An automated classification of the structure of protein loops. *J. Mol. Biol.* 266:814-830.
- Panchenko, A.R. 2003. Finding weak similarities between proteins by sequence profile comparison. *Nucl. Acids Res.* 31:683-689.
- Park, J., Karplus, K., Barrett, C., Hughey, R., Haussler, D., Hubbard, T., and Chothia, C. 1998. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.* 284:1201-1210.
- Pawlowski, K., Bierzynski, A., and Godzik, A. 1996. Structural diversity in a family of homologous proteins. *J. Mol. Biol.* 258:349-366.
- Pearl, F., Todd, A., Sillitoe, I., Dibley, M., Redfern, O., Lewis, T., Bennett, C., Marsden, R., Grant, A., Lee, D., Akpor, A., Maibaum, M., Harrison, A., Dallman, T., Reeves, G., Diboun, I., Addou, S., Lise, S., Johnston, C., Sillero, A., Thornton, J., and Orengo, C. 2005. The CATH Domain Structure Database and related resources Gene3D and DHS provide comprehensive domain family information for genome analysis. *Nucl. Acids Res.* 33:D247-D251.
- Pearson, W.R. 1994. Using the FASTA program to search protein and DNA sequence databases. *Methods Mol. Biol.* 24:307-331.
- Pearson, W.R. 2000. Flexible sequence similarity searching with the FASTA3 program package. *Methods Mol. Biol.* 132:185-219.
- Petrey, D. and Honig, B. 2005. Protein structure prediction: Inroads to biology. *Mol. Cell.* 20:811-819.
- Petrey, D., Xiang, Z., Tang, C.L., Xie, L., Gimpel, M., Mitros, T., Soto, C.S., Goldsmith-Fischman, S., Kernysky, A., Schlessinger, A., Koh, I.Y., Alexov, E., and Honig, B. 2003. Using multiple structure alignments, fast model building, and energetic analysis in fold recognition and homology modeling. *Proteins* 53:430-435.
- Pieper, U., Eswar, N., Braberg, H., Madhusudhan, M.S., Davis, F.P., Stuart, A.C., Mirkovic, N., Rossi, A., Marti-Renom, M.A., Fiser, A., Webb, B., Greenblatt, D., Huang, C.C., Ferrin, T.E., and Sali, A. 2004. MODBASE, a database of annotated comparative protein structure models, and associated resources. *Nucl. Acids Res.* 32:D217-D222.
- Pieper, U., Eswar, N., Davis, F.P., Braberg, H., Madhusudhan, M.S., Rossi, A., Marti-Renom, M., Karchin, R., Webb, B.M., Eramian, D., Shen, M.Y., Kelly, L., Melo, F., and Sali, A. 2006. MODBASE: A database of annotated comparative protein structure models and associated resources. *Nucl. Acids Res.* 34:D291-D295.
- Pietrokovski, S. 1996. Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucl. Acids Res.* 24:3836-3845.
- Pontius, J., Richelle, J., and Wodak, S.J. 1996. Deviations from standard atomic volumes as a quality measure for protein crystal structures. *J. Mol. Biol.* 264:121-136.
- Que, X., Brinen, L.S., Perkins, P., Herdman, S., Hirata, K., Torian, B.E., Rubin, H., McKerrow, J.H., and Reed, S.L. 2002. Cysteine proteinases from distinct cellular compartments are recruited to phagocytic vesicles by *Entamoeba histolytica*. *Mol. Biochem. Parasitol.* 119:23-32.
- Ring, C.S., Kneller, D.G., Langridge, R., and Cohen, F.E. 1992. Taxonomy and conformational analysis of loops in proteins. *J. Mol. Biol.* 224:685-699.
- Ring, C.S., Sun, E., McKerrow, J.H., Lee, G.K., Rosenthal, P.J., Kuntz, I.D., and Cohen, F.E. 1993. Structure-based inhibitor design by using protein models for the development of antiparasitic agents. *Proc. Natl. Acad. Sci. U.S.A.* 90:3583-3587.
- Rost, B. 1999. Twilight zone of protein sequence alignments. *Protein Eng.* 12:85-94.
- Rost, B. and Liu, J. 2003. The PredictProtein server. *Nucl. Acids Res.* 31:3300-3304.
- Rufino, S.D., Donate, L.E., Canard, L.H., and Blundell, T.L. 1997. Predicting the conformational class of short and medium size loops

- connecting regular secondary structures: Application to comparative modelling. *J. Mol. Biol.* 267:352-367.
- Rychlewski, L. and Fischer, D. 2005. LiveBench-8: The large-scale, continuous assessment of automated protein structure prediction. *Protein Sci.* 14:240-245.
- Rychlewski, L., Zhang, B., and Godzik, A. 1998. Fold and function predictions for Mycoplasma genitalium proteins. *Fold Des.* 3:229-238.
- Sadreyev, R. and Grishin, N. 2003. COMPASS: A tool for comparison of multiple protein alignments with assessment of statistical significance. *J. Mol. Biol.* 326:317-336.
- Sali, A. and Blundell, T.L. 1993. Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* 234:779-815.
- Sali, A. and Overington, J.P. 1994. Derivation of rules for comparative protein modeling from a database of protein structure alignments. *Protein Sci.* 3:1582-1596.
- Samudrala, R. and Moul, J. 1998. A graph-theoretic algorithm for comparative modeling of protein structure. *J. Mol. Biol.* 279:287-302.
- Sanchez, R. and Sali, A. 1997a. Advances in comparative protein-structure modelling. *Curr. Opin. Struct. Biol.* 7:206-214.
- Sanchez, R. and Sali, A. 1997b. Evaluation of comparative protein structure modeling by MODELLER-3. *Proteins* 1:50-58.
- Sanchez, R. and Sali, A. 1998. Large-scale protein structure modeling of the Saccharomyces cerevisiae genome. *Proc. Natl. Acad. Sci. U.S.A.* 95:13597-13602.
- Saqi, M.A., Russell, R.B., and Sternberg, M.J. 1998. Misleading local sequence alignments: Implications for comparative protein modelling. *Protein Eng.* 11:627-630.
- Sauder, J.M., Arthur, J.W., and Dunbrack, R.L. Jr. 2000. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins* 40:6-22.
- Schwarzenbacher, R., Godzik, A., Grzechnik, S.K., and Jaroszewski, L. 2004. The importance of alignment accuracy for molecular replacement. *Acta Crystallogr. D Biol. Crystallogr.* 60:1229-1236.
- Schwede, T., Kopp, J., Guex, N., and Peitsch, M.C. 2003. SWISS-MODEL: An automated protein homology-modeling server. *Nucl. Acids Res.* 31:3381-3385.
- Selzer, P.M., Chen, X., Chan, V.J., Cheng, M., Kenyon, G.L., Kuntz, I.D., Sakanari, J.A., Cohen, F.E., and McKerrow, J.H. 1997. Leishmania major: Molecular modeling of cysteine proteases and prediction of new nonpeptide inhibitors. *Exp. Parasitol.* 87:212-221.
- Sheng, Y., Sali, A., Herzog, H., Lahnstein, J., and Krilis, S.A. 1996. Site-directed mutagenesis of recombinant human beta 2-glycoprotein I identifies a cluster of lysine residues that are critical for phospholipid binding and anti-cardiolipin antibody activity. *J. Immunol.* 157:3744-3751.
- Shenkin, P.S., Yarmush, D.L., Fine, R.M., Wang, H.J., and Levinthal, C. 1987. Predicting antibody hypervariable loop conformation. I. Ensembles of random conformations for ringlike structures. *Biopolymers* 26:2053-2085.
- Shi, J., Blundell, T.L., and Mizuguchi, K. 2001. FUGUE: Sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.* 310:243-257.
- Sibanda, B.L., Blundell, T.L., and Thornton, J.M. 1989. Conformation of beta-hairpins in protein structures. A systematic classification with applications to modelling by homology, electron density fitting and protein engineering. *J. Mol. Biol.* 206:759-777.
- Sippl, M.J. 1990. Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. *J. Mol. Biol.* 213:859-883.
- Sippl, M.J. 1993. Recognition of errors in three-dimensional structures of proteins. *Proteins* 17:355-362.
- Sippl, M.J. 1995. Knowledge-based potentials for proteins. *Curr. Opin. Struct. Biol.* 5:229-235.
- Skolnick, J. and Kihara, D. 2001. Defrosting the frozen approximation: PROSPECTOR—a new approach to threading. *Proteins* 42:319-331.
- Smith, T.F. and Waterman, M.S. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147:195-197.
- Spahn, C.M., Beckmann, R., Eswar, N., Penczek, P.A., Sali, A., Blobel, G., and Frank, J. 2001. Structure of the 80S ribosome from Saccharomyces cerevisiae—tRNA-ribosome and subunit-subunit interactions. *Cell* 107:373-386.
- Srinivasan, N. and Blundell, T.L. 1993. An evaluation of the performance of an automated procedure for comparative modelling of protein tertiary structure. *Protein Eng.* 6:501-512.
- Sutcliffe, M.J., Haneef, I., Carney, D., and Blundell, T.L. 1987a. Knowledge based modelling of homologous proteins, Part I: Three-dimensional frameworks derived from the simultaneous superposition of multiple structures. *Protein Eng.* 1:377-384.
- Sutcliffe, M.J., Hayes, F.R., and Blundell, T.L. 1987b. Knowledge based modelling of homologous proteins, Part II: Rules for the conformations of substituted sidechains. *Protein Eng.* 1:385-392.
- Sutcliffe, M.J., Dobson, C.M., and Oswald, R.E. 1992. Solution structure of neuronal bungarotoxin determined by two-dimensional NMR spectroscopy: Calculation of tertiary structure using systematic homologous model building, dynamical simulated annealing, and restrained molecular dynamics. *Biochemistry* 31:2962-2970.
- Taylor, W.R., Flores, T.P., and Orengo, C.A. 1994. Multiple protein structure alignment. *Protein Sci.* 3:1858-1870.

- Thompson, J.D., Higgins, D.G., and Gibson, T.J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* 22:4673-4680.
- Thompson, J.D., Plewniak, F., and Poch, O. 1999. BALiBASE: A benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics* 15:87-88.
- Topham, C.M., McLeod, A., Eisenmenger, F., Overington, J.P., Johnson, M.S., and Blundell, T.L. 1993. Fragment ranking in modelling of protein structure. Conformationally constrained environmental amino acid substitution tables. *J. Mol. Biol.* 229:194-220.
- Topham, C.M., Srinivasan, N., Thorpe, C.J., Overington, J.P., and Kalsheker, N.A. 1994. Comparative modelling of major house dust mite allergen Der p I: Structure validation using an extended environmental amino acid propensity table. *Protein Eng.* 7:869-894.
- Unger, R., Harel, D., Wherland, S., and Sussman, J.L. 1989. A 3D building blocks approach to analyzing and predicting structure of proteins. *Proteins* 5:355-373.
- Vakser, I.A. 1995. Protein docking for low-resolution structures. *Protein Eng.* 8:371-377.
- van Gelder, C.W., Leusen, F.J., Leunissen, J.A., and Noordik, J.H. 1994. A molecular dynamics approach for the generation of complete protein structures from limited coordinate data. *Proteins* 18:174-185.
- van Vlijmen, H.W. and Karplus, M. 1997. PDB-based protein loop prediction: Parameters for selection and methods for optimization. *J. Mol. Biol.* 267:975-1001.
- Vernal, J., Fiser, A., Sali, A., Muller, M., Cazzulo, J.J., and Nowicki, C. 2002. Probing the specificity of a trypanosomal aromatic alpha-hydroxy acid dehydrogenase by site-directed mutagenesis. *Biochem. Biophys. Res. Commun.* 293:633-639.
- von Ohlsen, N., Sommer, I., and Zimmer, R. 2003. Profile-profile alignment: A powerful tool for protein structure prediction. *Pac. Symp. Biocomput.* 2003:252-263.
- Vriend, G. 1990. WHAT IF: A molecular modeling and drug design program. *J. Mol. Graph* 8:52-56, 29.
- Wang, G. and Dunbrack, R.L. Jr. 2004. Scoring profile-to-profile sequence alignments. *Protein Sci.* 13:1612-1626.
- Wolf, E., Vassilev, A., Makino, Y., Sali, A., Nakatani, Y., and Burley, S.K. 1998. Crystal structure of a GCN5-related N-acetyltransferase: *Serratia marcescens* aminoglycoside 3-N-acetyltransferase. *Cell* 94:439-449.
- Worley, K.C., Culpepper, P., Wiese, B.A., and Smith, R.F. 1998. BEAUTY-X: Enhanced BLAST searches for DNA queries. *Bioinformatics* 14:890-891.
- Wu, G., Fiser, A., ter Kuile, B., Sali, A., and Muller, M. 1999. Convergent evolution of *Trichomonas vaginalis* lactate dehydrogenase from malate dehydrogenase. *Proc. Natl. Acad. Sci. U.S.A.* 96:6285-6290.
- Xiang, Z., Soto, C.S., and Honig, B. 2002. Evaluating conformational free energies: The colony energy and its application to the problem of loop prediction. *Proc. Natl. Acad. Sci. U.S.A.* 99:7432-7437.
- Xu, J., Li, M., Kim, D., and Xu, Y. 2003. RAPTOR: Optimal protein threading by linear programming. *J. Bioinform. Comput. Biol.* 1:95-117.
- Xu, L.Z., Sanchez, R., Sali, A., and Heintz, N. 1996. Ligand specificity of brain lipid-binding protein. *J. Biol. Chem.* 271:24711-24719.
- Ye, Y., Jaroszewski, L., Li, W., and Godzik, A. 2003. A segment alignment approach to protein comparison. *Bioinformatics* 19:742-749.
- Yona, G. and Levitt, M. 2002. Within the twilight zone: A sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.* 315:1257-1275.
- Zheng, Q., Rosenfeld, R., Vajda, S., and DeLisi, C. 1993. Determining protein loop conformation using scaling-relaxation techniques. *Protein Sci.* 2:1242-1248.
- Zhou, H. and Zhou, Y. 2002. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Sci.* 11:2714-2726.
- Zhou, H. and Zhou, Y. 2004. Single-body residue-level knowledge-based energy score combined with sequence-profile and secondary structure information for fold recognition. *Proteins* 55:1005-1013.
- Zhou, H., and Zhou, Y. 2005. Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins* 58:321-328.

Internet Resources

<http://www.salilab.org/modeller>

Eswar, N., Madhusudhan, M.S., Marti-Renom, M.A., and Sali, A. 2005. MODELLER, A Protein Structure Modeling Program, Release 8v.2.

Contributed by Narayanan Eswar, Ben Webb, Marc A. Marti-Renom, M.S. Madhusudhan, David Eramian, Min-yi Shen, Ursula Pieper, and Andrej Sali
University of California at San Francisco
San Francisco, California

Using VMD: An Introductory Tutorial

UNIT 5.7

Jen Hsin,^{1,2} Anton Arkhipov,^{1,2} Ying Yin,^{1,2} John E. Stone,² and Klaus Schulten^{1,2}

¹Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois

²Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, Illinois

ABSTRACT

VMD (Visual Molecular Dynamics) is a molecular visualization and analysis program designed for biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. This unit will serve as an introductory VMD tutorial. We will present several step-by-step examples of some of VMD's most popular features, including visualizing molecules in three dimensions with different drawing and coloring methods, rendering publication-quality figures, animating and analyzing the trajectory of a molecular dynamics simulation, scripting in the text-based Tcl/Tk interface, and analyzing both sequence and structure data for proteins. *Curr. Protoc. Bioinform.* 24:5.7.1-5.7.48. © 2008 by John Wiley & Sons, Inc.

Keywords: molecular modeling • molecular dynamics visualization • interactive visualization • animation

INTRODUCTION

VMD (Visual Molecular Dynamics; Humphrey et al., 1996) is a molecular visualization and analysis program designed for biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. It is developed by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. Among molecular graphics programs, VMD is unique in its ability to efficiently operate on multi-gigabyte molecular dynamics trajectories, its interoperability with a large number of molecular dynamics simulation packages, and its integration of structure and sequence information.

Key features of VMD include methods: (1) general 3D molecular visualization with extensive drawing and coloring methods (e.g., see Fig. 5.7.1); (2) extensive atom selection syntax for choosing subsets of atoms for display; (3) visualization of dynamic molecular data; (4) visualization of volumetric data; (5) support for most molecular data file formats; (6) no limits on the number of atoms, molecules, or trajectory frames, except available memory; (7) molecular analysis commands; (8) rendering high-resolution, publication-quality molecule images; (9) movie making capability; (10) building and preparing systems for molecular dynamics simulations; (11) interactive molecular dynamics simulations; (12) extensions to the Tcl/Python scripting languages; and (13) extensible source code written in C and C++.

This unit will serve as an introductory VMD tutorial. It is impossible to cover all of VMD's capabilities in one unit; instead, we will present several step-by-step examples of VMD's basic features. Topics covered in this tutorial include visualizing molecules in three dimensions with different drawing and coloring methods, rendering publication-quality figures, animating and analyzing the trajectory of a molecular dynamics simulation, scripting in the text-based Tcl/Tk interface, and analyzing both sequence and structure data for proteins.

Modeling
Structure from
Sequence

5.7.1

Supplement 24

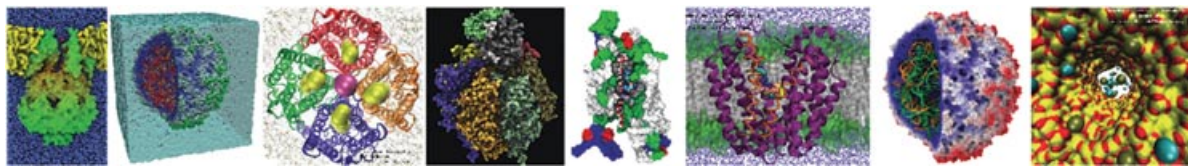


Figure 5.7.1 Example renderings made with VMD (Cruz-Chu et al., 2006; Freddolino et al., 2006; Yin et al., 2006; Yu et al., 2006; Sotomayor et al., 2007; Wang et al., 2007). For the color version of this figure go to <http://www.currentprotocols.com>.

DOWNLOADING VMD

Before starting, the current version of VMD needs to be downloaded. This tutorial was written for VMD version 1.8.6. VMD supports all major computer platforms and can be downloaded from the VMD homepage <http://www.ks.uiuc.edu/Research/vmd>. Follow the instructions online to install. Once VMD is installed, to start VMD if using Mac OS X, double-click on the VMD application icon in the Applications directory; if using Linux and SUN, type `vmd` in a terminal window, or if using Windows, select → Start Programs → VMD.

When VMD starts, by default three windows will open: the VMD Main window, the OpenGL Display window, and the VMD Console window (or a Terminal window on a Mac). To end a VMD session, go to the VMD Main window, and choose File → Quit. You can also quit VMD by closing the VMD Console window or the VMD Main window.

TOPICS AND FILES

This unit contains six sections. Each section acts as an independent tutorial for a specific topic (Working with a Single Molecule, Trajectories and Movie Making, Scripting in VMD, Working with Multiple Molecules, Comparing Protein Structures and Sequences with the MultiSeq Plugin, and Data Analysis in VMD). For readers with no prior experience with VMD, we suggest they work through the sections in the order they are presented. Readers already familiar with the basics of VMD may selectively pursue sections of their interest. Several files have been prepared to accompany this tutorial. You need to download these files at <http://www.currentprotocols.com>.

WORKING WITH A SINGLE MOLECULE

In this section, the basic functions of VMD will be introduced, starting with loading a molecule, displaying the molecule, and rendering publication-quality molecule images. This section uses the protein ubiquitin as an example molecule. Ubiquitin is a small protein responsible for labeling proteins for degradation, and is found in all eukaryotes with nearly identical sequences and structures.

Necessary Resources

Hardware

Computer

Software

VMD, and an image-displaying program

Files

1ubq.pdb, which can be downloaded at <http://www.currentprotocols.com>

Loading and Displaying the Molecule

A VMD session usually starts with loading structural information of a molecule into VMD. When VMD loads a molecule, it accesses the information about the names and coordinates of the atoms. Then, one can explore various VMD visualization features to get a nice view of the loaded molecule.

Loading a molecule

The first step is to load the molecule. The pdb file, 1ubq.pdb (Vijay-Kumar et al., 1987) that contains the atomic coordinates of ubiquitin will be loaded.

1. Start a VMD session. In the VMD Main window, choose File → New Molecule... (Fig. 5.7.2A). The Molecule File Browser window (Fig. 5.7.2B) will appear on the screen.
2. Use the Browse... (Fig. 5.7.2C) button to find the file 1ubq.pdb. When the file is selected, you will be back in the Molecule File Browser window. In order to actually load the file, press Load (Fig 5.7.2D).
3. Now, ubiquitin is shown in the OpenGL Display window. Close the Molecule File Browser window at any time.

VMD can download a pdb file from the Protein Data Bank (<http://www.pdb.org>) if a network connection is available. Just type the four letter code of the protein in the File Name text entry of the Molecule File Browser window and press the Load button. VMD will download it automatically.

Displaying the molecule

In order to see the 3D structure of our protein, the mouse will be used in multiple modes to change the viewpoint. VMD allows users to rotate, scale, and translate the viewpoint of the molecule.

4. In the OpenGL Display, press the left mouse button down and move the mouse. Explore what happens. This is the rotation mode of the mouse and allows for rotation of the molecule around an axis parallel to the screen (Fig. 5.7.3A).

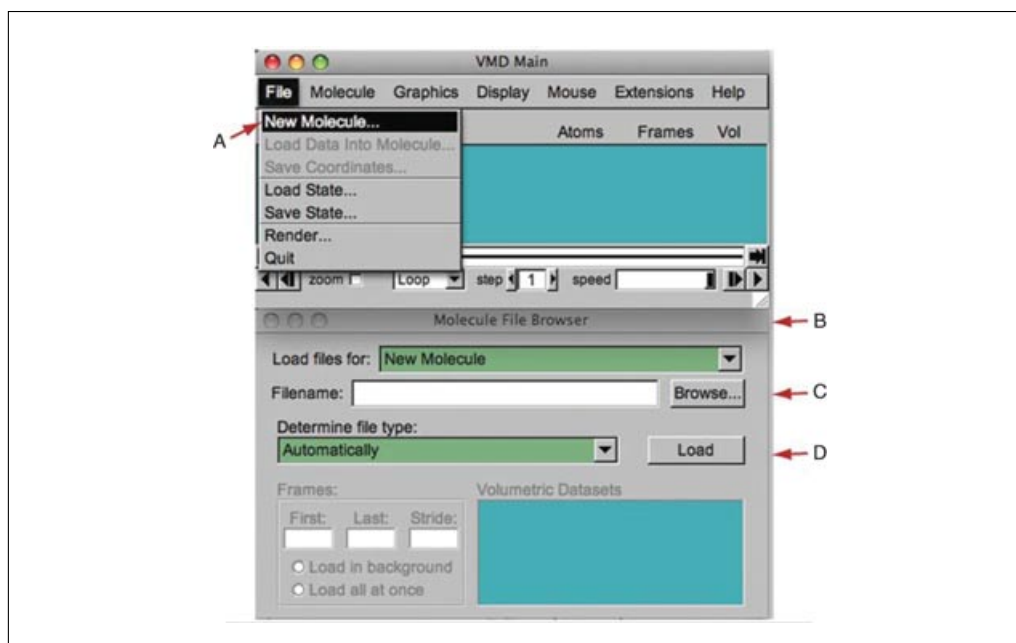


Figure 5.7.2 Loading a molecule.

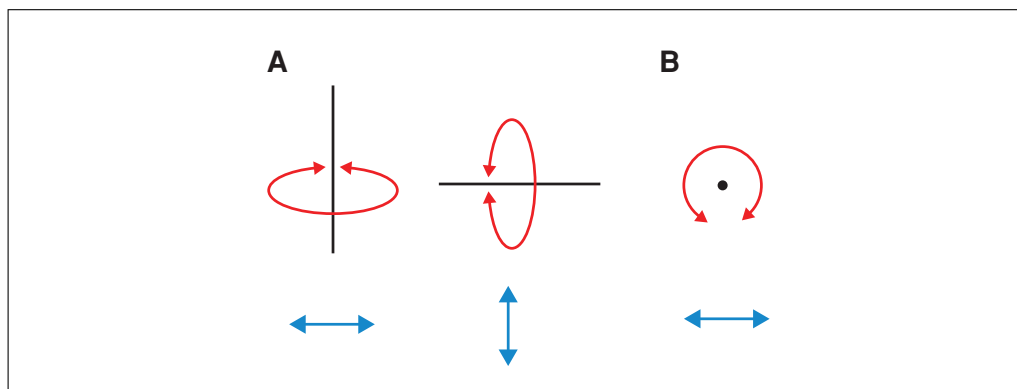


Figure 5.7.3 Rotational modes. (A) Rotation axes when holding down the left mouse key. (B) The rotation axes when holding down the right mouse key. For the color version of this figure go to <http://www.currentprotocols.com>.

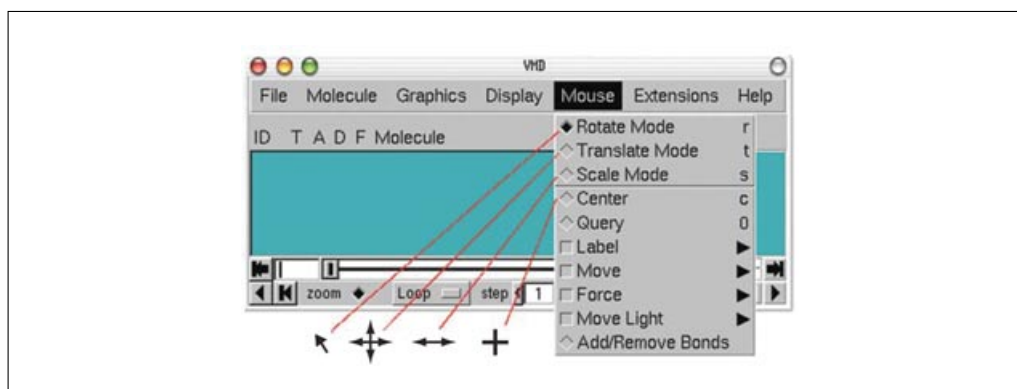


Figure 5.7.4 Mouse modes and their characteristic cursors.

5. Holding down the right mouse button and repeating the previous step will cause rotation around an axis perpendicular to the screen (Fig. 5.7.3B).

For Mac users who have a single-button mouse or a trackpad, the right mouse button is equivalent to holding down the command key while pressing the mouse/trackpad button.

6. In the VMD Main window, look at the Mouse menu (Fig. 5.7.4). Here, the user is able to switch the mouse mode from Rotation to Translation or Scale modes.
7. Choose the Translation mode and go back to the OpenGL Display. It is now possible to move the molecule around when you hold the left mouse button down.
8. Go back to the Mouse menu and choose the Scale mode this time. This will allow the user to zoom in or out by moving the mouse horizontally while holding down the left mouse button.

It should be noted that these actions performed with the mouse only change the viewpoint and do not change the actual coordinates of the molecule's atoms.

Also note that each mouse mode has its own characteristic cursor and its own shortcut key (r: Rotate, t: Translate, s: Scale). When the OpenGL Display window is the active window, these shortcut keys can be used instead of the Mouse menu to change the mouse mode.

Another useful option is the Mouse → Center menu item. It allows you to specify the point around which rotations are done.

9. Select the Center menu item and pick one atom at one of the ends of the protein; the cursor should display a cross.

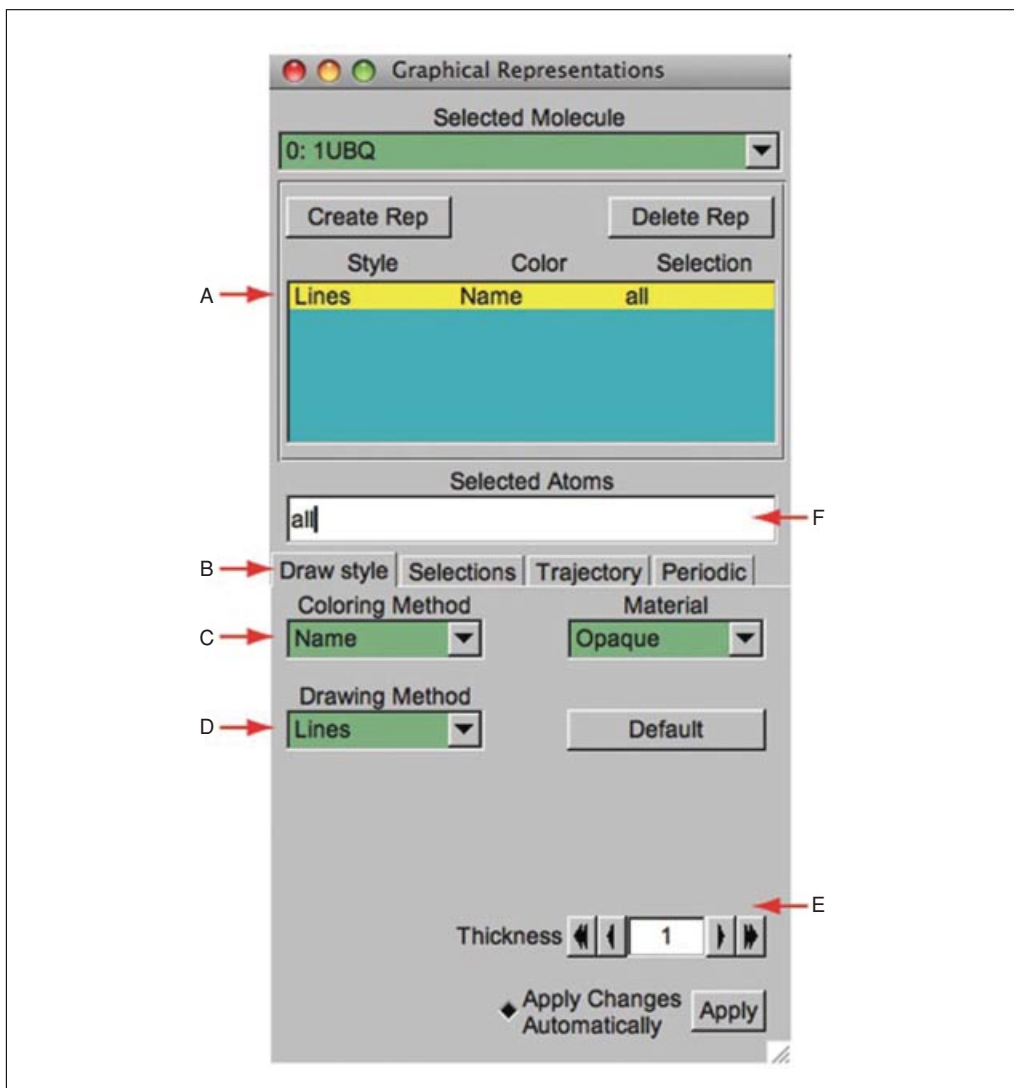


Figure 5.7.5 The Graphical Representations window. (A) List of representations, (B) the tabs for Draw Style, Selections, Trajectory, and Periodic, (C) Coloring Method pull-down menu, (D) Drawing Method pull-down menu, (E) user-adjustable parameters for different drawing methods, and (F) selection text entry box.

10. Now, press $\mathcal{R}$, and rotate the molecule with the mouse and see how the molecule moves around the selected point.
11. In the VMD Main window, select the Display → Reset View menu item to return to the default view. You can also reset the view by pressing the “=” key when you are in the OpenGL Display window.

Graphical representations

VMD can display molecules in various ways by setting the Graphical Representations window shown in Figure 5.7.5. Each representation is defined by four main parameters: the selection of atoms included in the representation, the drawing style, the coloring method, and the material. The selection determines which part of the molecule is drawn; the drawing method defines which graphical representation is used; the coloring method gives the color of each part of the representation; and the material determines the effects of lighting, shading, and transparency on the representation. Let us first explore different drawing styles.

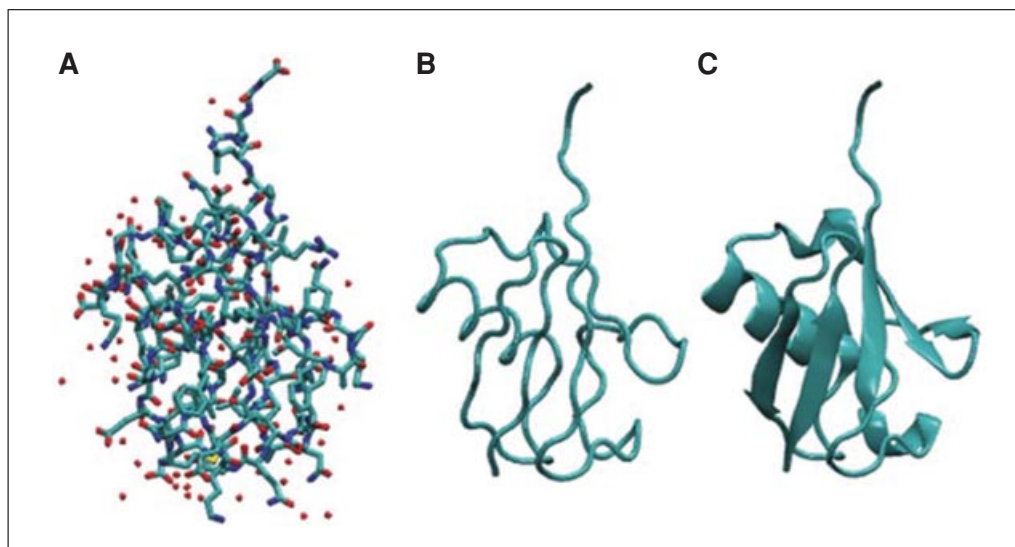


Figure 5.7.6 (A) Licorice, (B) Tube, and (C) NewCartoon representations of ubiquitin. For the color version of this figure go to <http://www.currentprotocols.com>.

Exploring different drawing styles

12. In the VMD Main window, choose the Graphics → Representations... menu item. A window called Graphical Representations will appear and the current default representation will be highlighted in yellow (Fig. 5.7.5A).
13. In the Draw Style tab (Fig. 5.7.5B), change the style (Fig. 5.7.5D) and color (Fig. 5.7.5C) of the representation. Here, we will focus on the drawing style (the default is Lines).
14. Each Drawing Method has its own parameters. For instance, change the thickness of the lines by using the controls on the lower right-hand-side corner (Fig. 5.7.5E) of the Graphical Representations window.
15. Click on the Drawing Method (Fig. 5.7.5D) to see a list of options. Choose VDW (van der Waals); each atom is now represented by a sphere scaled to its van der Waals radius, allowing the user to see the volumetric distribution of the protein.
16. When choosing VDW as the drawing method, two new controls will show up in the lower right-hand-side corner. Use these controls to change the Sphere Scale to 0.5 and the Sphere Resolution to 13. Note that the higher the resolution, the slower the display of the molecule will be.
17. Press the Default button. This returns the screen to the default properties of the chosen drawing method.

Other popular representations include CPK and Licorice. In CPK, like in old chemistry ball and stick kits, each atom is represented by a sphere and each bond is represented by a thin cylinder (radius and resolution of both the sphere and the cylinder can be modified). The Licorice drawing method also represents each atom as a sphere and each bond as a cylinder, but the sphere and the cylinder have the same radii.

Using the Tube style drawing method

The previous representations visualize micromolecular details of the protein by displaying every single atom. More general structural properties can be demonstrated better by using more abstract drawing methods.

18. Choose the Tube style under Drawing Method, which shows the backbone of the protein. Set the Radius to 0.8. The result should be similar to Figure 5.7.6.

Using the NewCartoon drawing method

The last drawing method described here is NewCartoon. It gives a simplified representation of a protein based on its secondary structure. Helices are drawn as coiled ribbons, β -sheets as solid, flat arrows, and all other structures as a tube. This is probably the most popular drawing method to view the overall architecture of a protein.

19. In the Graphical Representations window, choose Drawing Method $\rightarrow$ NewCartoon. The helices, β -sheets, and coils of the protein can now be easily identified.

Ubiquitin has three and one half turns of α -helix (residues 23 to 34, three of them hydrophobic), one short piece of 3_{10} -helix (residues 56 to 59) and a mixed β -sheet with five strands (residues 1 to 7, 10 to 17, 40 to 45, 48 to 50, and 64 to 72), and seven reverse turns. VMD uses the program STRIDE (Frishman and Argos, 1995) to compute the secondary structure according to a heuristic algorithm.

Exploring different coloring methods

In this series of steps, different coloring methods are explored.

20. In the Graphical Representations window, the default coloring method is Coloring Method $\rightarrow$ Name. In this coloring method, choose a drawing method that shows individual atoms: each atom will have a different color, i.e., O is red, N is blue, C is cyan, and S is yellow.
21. Choose Coloring Method $\rightarrow$ ResType (Fig. 5.7.5C). This allows nonpolar residues (white) to be distinguished from basic residues (blue), acidic residues (red), and polar residues (green).
22. Select Coloring Method $\rightarrow$ Structure (Fig. 5.7.5C) and confirm that the NewCartoon representation displays colors consistent with secondary structure.

Displaying different selections

To display only parts of the molecule of interest, one can specify their selection in the Graphical Representations window (Fig. 5.7.5F).

23. In the Graphical Representations window, there is a Selected Atoms text entry (Fig. 5.7.5F). Delete the word `all`, type `helix`, and press the Apply button or hit the Enter/return key (remember to do this whenever a selection is changed). VMD will show just the helices present in the molecule.
24. In the Graphical Representations window, choose the Selections tab (Fig. 5.7.7A). In the section Singlewords (Fig. 5.7.7B), a list of possible selections that can be entered is provided.

Combinations of Boolean operators can also be used when writing a selection.

25. In order to see the molecule without helices and β -sheets, type the following in the Selected Atoms field: `(not helix) and (not betasheet)`. Remember to press the Apply button or hit the Enter/return key.
26. In the section Keyword (Fig. 5.7.7C) of the Selections tab, the properties that can be used to select parts of a molecule are listed along with their possible values. Look at possible values of the keyword “resname” (Fig. 5.7.7D).
27. Display all the lysines and glycines present in the protein by typing `(resname LYS) or (resname GLY)` in the Selected Atoms field.

Lysines play a fundamental role in the configuration of polyubiquitin chains.

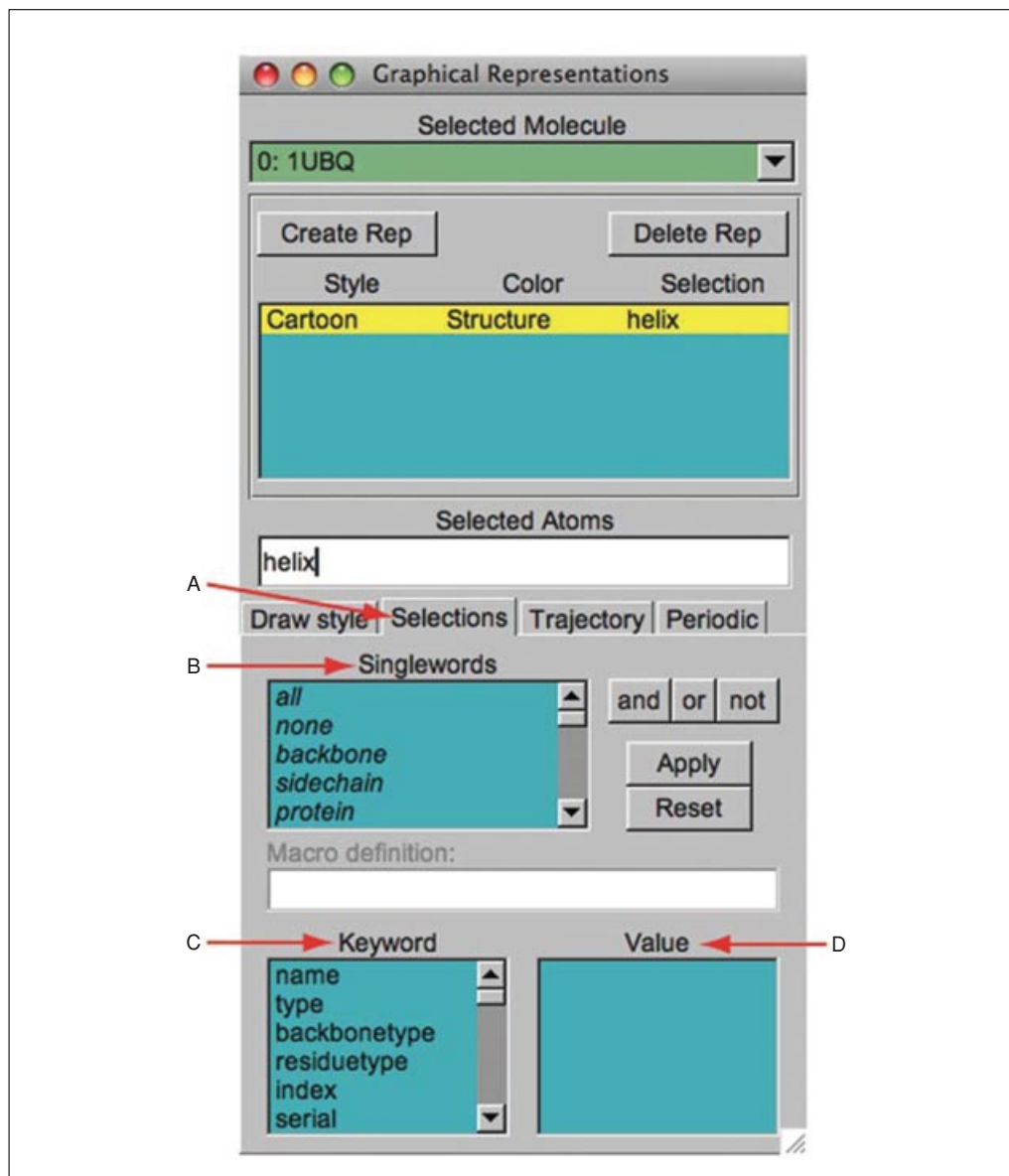


Figure 5.7.7 Graphical Representations window and the (A) Selections tab, (B) list of Singlewords, (C) list of Keywords, and (D) Value box that displays possible choices for a given keyword.

28. Change the current representation's Drawing Method to CPK and the Coloring Method to ResName in the Draw style tab. In the screen, the different lysines and glycines will be visible.
29. In the Selected Atoms text field, entry type `water`. Choose Coloring Method → Name. The 58 water molecules present in the system now appear (in fact only their oxygen atoms).
30. In order to see which water molecules are closer to the protein, use the command `within`. Type `water` and `within 3` of `protein` for Selected Atoms in the text field.

This selects all the water molecules that are within a distance of 3 Å of the protein.

31. Finally, try typing in the Selected Atoms field the selections shown in the first column of Table 5.7.1. Each of these selections will show the protein or part of the protein as explained in the second column of Table 5.7.1.

Table 5.7.1 Examples of Atom Selections

Selection	Action
Protein	Shows the protein
resid 1	The first residue
(resid 1 76) and (not water)	The first and last residues
(resid 23 to 34) and (protein)	The α -helix

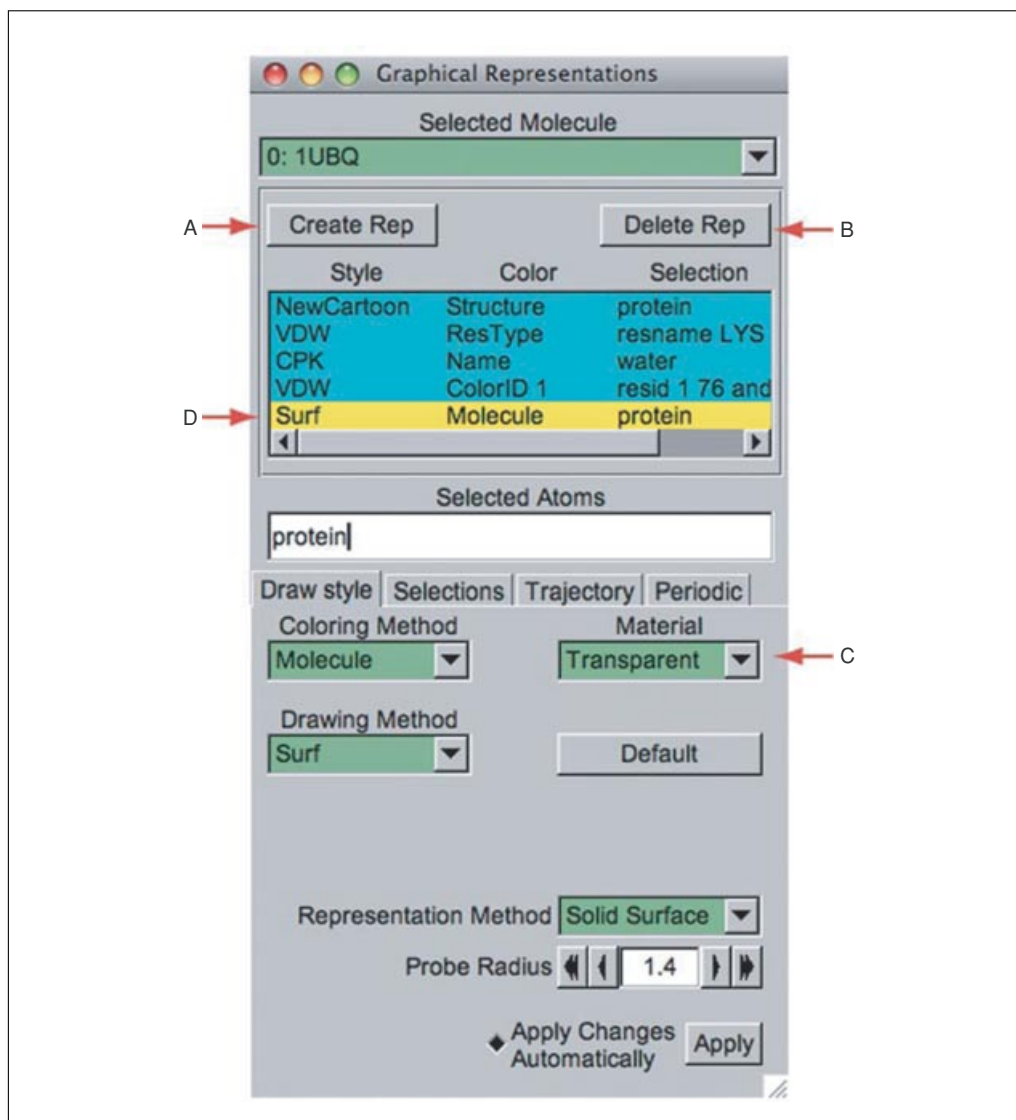


Figure 5.7.8 Multiple Representations of ubiquitin. Representations can be either created or deleted using the (A) Create Rep and (B) Delete Rep buttons. Screen also shows (C) the Material pull-down menu and (D) list of representations.

Creating multiple representations

The button Create Rep (Fig. 5.7.8A) in the Graphical Representations window allows creation of multiple representations. Therefore, users can have a mixture of different selections with different styles and colors, all displayed at the same time.

32. For the current representation, in the Selected Atoms field type `protein`, set the Drawing Method to NewCartoon and the Coloring Method to Structure in the Draw style tab.

Table 5.7.2 Examples of Representations

Selection	Coloring method	Drawing method
Water	Name	CPK
resid 1 76 and name CA	ColorID 1	VDW

33. Press the Create Rep button (Fig. 5.7.8A). A new representation will be created.
34. Modify the new representation to get VDW as the Drawing Method, ResType as the Coloring Method, and `resname LYS` as the current selection.
35. Repeating the previous procedure, create the following two new representations in Table 5.7.2. These two representations show water molecules and the C α atoms of the first and last residues of the protein.
36. Create the last representation by pressing the Create Rep button again. Select Drawing Method $\rightarrow$ Surf for drawing method, Coloring Method $\rightarrow$ Molecule for coloring method, and type `protein` in the Selected Atoms field. For this last representation, choose Transparent in the Material pull-down menu (Fig. 5.7.8C). This representation shows the protein's volumetric surface in transparent.

Note that you can select and modify different representations you have created by clicking on a representation to highlight it in yellow. Also, each representation can be switched on/off by double-clicking on it. To delete a representation, highlight it and then click on the Delete Rep button (Fig. 5.7.8B). At the end of this section, the Graphical Representations window should look like Figure 5.7.8.

Sequence viewer extension

When dealing with a protein for the first time, it is very useful to be able to find and display different amino acids quickly. The sequence viewer extension allows viewing of the protein sequence, as well as to easily pick and display one or more residues of interest.

37. In the VMD Main window, choose the Extension $\rightarrow$ Analysis $\rightarrow$ Sequence Viewer menu item. A window (Fig. 5.7.9A) with a list of the amino acids (Fig. 5.7.9E) and their properties (Figs. 5.7.9B through 5.7.9C) will appear on the screen.
38. With the mouse, try clicking on different residues in the list (Fig. 5.7.9E) and see how they are highlighted. In addition, the highlighted residue will appear in the OpenGL Display window in yellow and rendered in the bond drawing method, so its location within the protein can be visualized easily.
39. Use the Zoom controls (Fig. 5.7.9F) to display the entire list of residues in the window. This is particularly useful for larger proteins.
40. Pick multiple residues by holding the shift key and clicking on the mouse button (Fig. 5.7.9E).
41. Look at the Graphical Representations window; a new representation with the residues that have been selected using the Sequence Viewer Extension should be shown. Modify, hide, or delete this representation similar to the steps described above.

*Information about residues is color-coded (Fig. 5.7.9D) in columns and obtained from STRIDE. The B-value column (Fig. 5.7.9B) shows the B-value field (temperature factor) often provided in *pdb* files. The "struct" column shows secondary structure (Fig. 5.7.9D), where each letter corresponds to a secondary structure, listed in Table 5.7.3.*

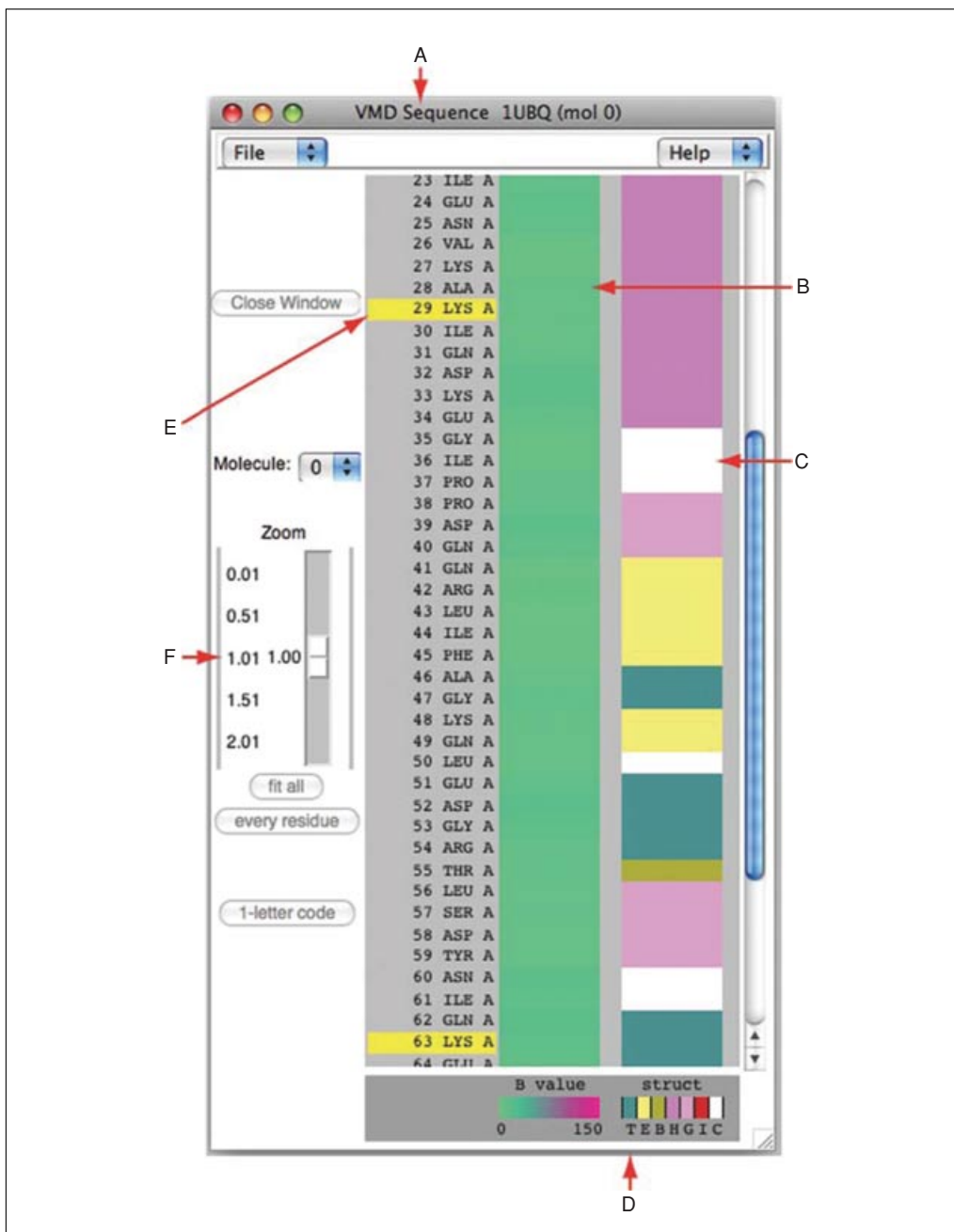


Figure 5.7.9 (A) The VMD Sequence window displays properties of the protein sequence, including (B) the B-value and (C) the secondary structure, denoted by (D) the color codes. (E) The list of residues is displayed, with the selected residues highlighted in yellow. (F) Zoom controls are also shown in the window. For the color version of this figure go to <http://www.currentprotocols.com>.

Saving your work

The viewpoints and representations created using VMD can be saved as a VMD state. This VMD state contains all the information needed to reproduce the same VMD session.

42. Go to the OpenGL Display window; use the mouse to find a nice view of the protein. We will save this viewpoint using the VMD ViewMaster.
43. In the VMD Main window (Fig. 5.7.2), select Extension → Visualization → ViewMaster. This will open the VMD ViewMaster window.
44. In the VMD ViewMaster window, click on the Create New button. The OpenGL Display viewpoint has now been saved.

Table 5.7.3 Secondary Structure Codes Used by STRIDE

Letter code	Secondary structure
T	Turn
E	Extended conformation (β -sheets)
B	Isolated bridge
H	Alpha helix
G	3-10 helix
I	Pi helix
C	Coil

45. Go back to the OpenGL Display window and use the mouse to find another nice view. If desired, you can add/delete/modify a representation in the Graphical Representations window. When a good view has been found, save it by returning to the VMD ViewMaster window and clicking on the Create New button.
46. Create as many views as desired by repeating the previous step. All of the viewpoints are displayed as thumbnails in the VMD ViewMaster window. A previously saved viewpoint can be opened by clicking on its thumbnail.
47. To save the entire VMD session, in the VMD Main window, choose the File $\rightarrow$ Save State menu item. Type an appropriate name (e.g., `myfirststate.vmd`) and save it.

The VMD state file `myfirststate.vmd` contains all the information needed to restore a VMD session, including the viewpoints and the representations.

To load a saved VMD state, start a new VMD session and in the VMD Main window choose File $\rightarrow$ Load State.

48. Quit VMD.

BASIC PROTOCOL 2

The Basics of VMD Figure Rendering

One of VMD's many strengths is its ability to render high-resolution, publication-quality molecule images. In this section, we will introduce some basic concepts of figure rendering in VMD.

Setting the display background

Before rendering a figure, make sure that the OpenGL Display background is set up the way you want. Nearly all aspects of the OpenGL Display are user-adjustable, including the background color.

1. Start a new VMD session (Basic Protocol 1) and load the `1ubq.pdb` file.
2. In the VMD Main window, choose Graphics $\rightarrow$ Colors. . . . The Color Controls window should show up. Look through the Categories list. All display colors, for example, the colors of different atoms when colored by name, are set here.
3. In Categories, select Display. In Names, select Background. Finally, choose "8 white" in Colors. The OpenGL Display should now have a white background.
4. When making a figure, we often do not want to include the axes. To turn off the axes, select Display $\rightarrow$ Axes $\rightarrow$ Off in the VMD Main window.

Increasing geometric resolution

All VMD objects are drawn with an adjustable resolution, allowing users to balance fineness of detail with drawing speed.

5. Open the Graphical Representation window via Graphics → Representations. . . in the VMD Main menu. Modify the default representation to show just the protein, and display it using the VDW drawing method.
6. Zoom in on one or two of the atoms by using Mouse Scale → Mode (shortcut s).

You might notice that as you zoom into an atom closer and closer, the atom might be cut off by an invisible clipping plane, which makes it difficult to focus on just one atom. This is an OpenGL feature. You can move the clipping plane closer to you by doing the following: switch your mouse mode to the Translate mode, either by pressing the shortcut key “t” in the OpenGL window or by selecting Mouse → Translate Mode, and dragging your mouse in the OpenGL window while holding down the right mouse key. You can now move the clipping plane closer to you, or away from you. If this does not work, here is an alternative way: in the VMD Main window, choose Display → Display Settings. . . ; in the Display Settings window that shows up you can see that many OpenGL options are adjustable; decrease the value for Near Clip, which will move the OpenGL clipping closer, allowing you to zoom in on individual atoms without clipping them off.

7. Notice that with the default resolution setting, the “spherical” atoms are not looking very spherical. In the Graphical Representations window, click on the representation you set up before for the protein to highlight it in yellow. Try adjusting the Sphere Resolution setting to something higher, and see what a difference it can make (Fig. 5.7.10).

Most of the drawing methods have a geometric resolution setting. Try a few different drawing methods and see how their resolutions can be easily increased. When producing images, the resolution can be raised until it stops making a visible difference.

Colors and materials

8. There is a Material menu in the Graphical Representations window (which by default is set to Opaque material). Choose the protein representation you made before, and experiment with the different materials in the Material menu.

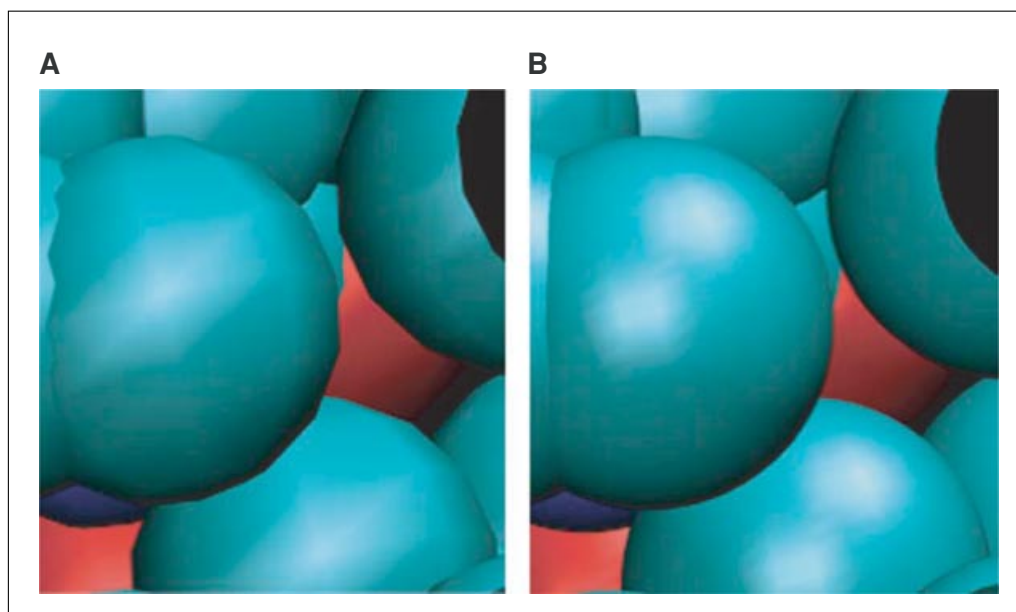


Figure 5.7.10 The effect of the resolution setting. (A) Low resolution: Sphere Resolution set to 8. (B) High resolution: Sphere Resolution set to 28.

9. Besides the predefined materials in the Material menu, VMD also allows users to create their own materials. To make a new material, in the VMD Main window choose Graphics → Materials. . .
 10. In the Materials window that appears, you will see a list of the materials you just tried out, and their adjustable settings. Click the Create New button. A new material, Material 12, will be created. Give it the settings listed in Table 5.7.4.
 11. Go back to the Graphical Representations window. In the Material menu, Material 12 is now on the list. Try using Material 12 for a representation and see what it looks like. You can also rename the materials in the Material menu.
- Now is a good time to try out the GLSL Render Mode, if your computer supports it. In the VMD Main window, choose Display → Rendermode → GLSL. This mode uses your 3D graphics card to render the scene with real-time ray-tracing of spheres and alpha-blended transparency, and can improve the visualization of transparent materials. See Figure 5.7.11 for example renderings made in GLSL mode.*
12. If your computer supports GLSL Render Mode, you can try to reproduce Figure 5.7.11. First, turn on the GLSL rendering mode by selecting Display → Rendermode → GLSL in the VMD Main window.
 13. Modify Material 12 to be more transparent by entering the values listed in Table 5.7.5 in the Materials window.

Table 5.7.4 Example of a User-Defined Material

Setting	Value
Ambient	0.30
Diffuse	0.30
Specular	0.90
Shininess	0.50
Opacity	0.95

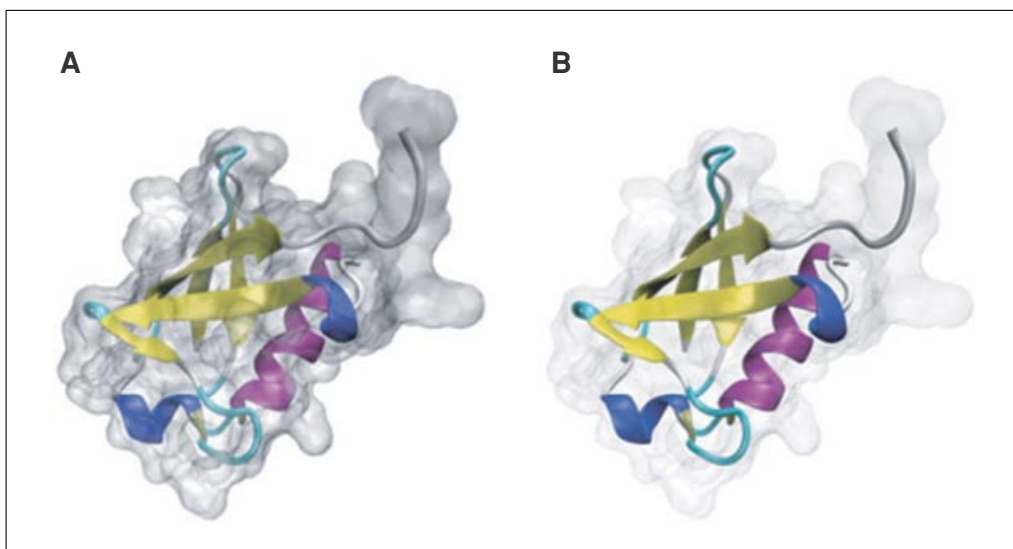


Figure 5.7.11 Examples of different material settings. (A) The default transparent material, rendered in GLSL mode. (B) A user-defined material with high transparency, also rendered in GLSG mode. For the color version of this figure go to <http://www.currentprotocols.com>.

Table 5.7.5 Example of a More Transparent Material

Setting	Value
Ambient	0.30
Diffuse	0.50
Specular	0.87
Shininess	0.85
Opacity	0.11

Table 5.7.6 Example of Representations Drawn with Different Materials

Selection	Coloring method	Drawing method	Material
protein	Structure	NewCartoon	Opaque
protein	ColorID→8 white	Surf	Material 12

14. Hide all of the current representations and create the two representations listed in Table 5.7.6.

Depth perception

Since the molecular systems are three-dimensional, VMD has multiple ways of representing the third dimension. In this section, how to use VMD to enhance or hide depth perception is discussed.

15. The first thing to consider is the projection mode. In the VMD Main window, click the Display menu. Here, we can choose either Perspective or Orthographic in the drop-down menu. Try switching between Perspective or Orthographic projection modes and see the difference (Fig. 5.7.12).

In perspective mode, things closer to the camera appear larger. Perspective projection provides strong size-based visual depth cues, but the displayed image will not preserve scale relationships or parallelism of lines, and objects very close to the camera may appear distorted. Orthographic projection preserves scale and parallelism relationships between objects in the displayed image, but greatly reduces depth perception. Hence, orthographic mode tends to be more useful for analysis, because alignment is easy to see, while perspective mode is often used for producing figures and stereo images.

Another way VMD can represent depth is through so-called “depth cueing.” Depth cueing is used to enhance three-dimensional perception of molecular structures, particularly with orthographic projections.

16. Choose Display → Depth Cueing in the VMD Main window.

When depth cueing is enabled, objects further from the camera are blended into the background. Depth cueing settings are found in Display → Display Settings. . . . Here one can choose the functional dependence of the shading on distance, as well as some parameters for this function. To see the depth cueing effect better, you might want to hide the representation with the Surf drawing method.

17. Finally, VMD can also produce stereo images. In the VMD Main window, look at the Display → Stereo menu, showing many different choices. Choose SideBySide (remember to return to Perspective mode for a better result). The result should look like Figure 5.7.13.
18. Turn off stereo image by selecting Display → Stereo → Off in the VMD Main window. Also, turn off depth cueing by unselecting the Display → Depth Cueing checkbox in the VMD Main window.

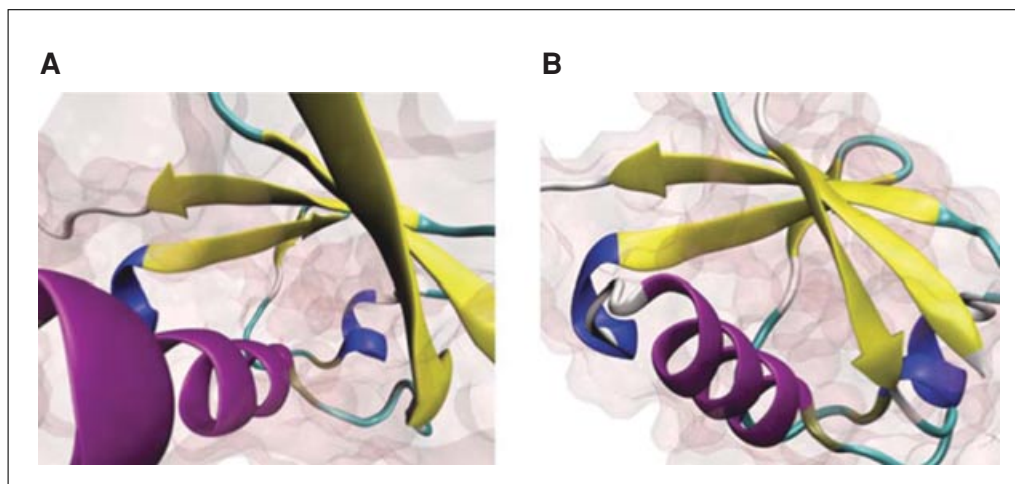


Figure 5.7.12 Comparison of the (A) perspective and (B) orthographic projection modes. For the color version of this figure go to <http://www.currentprotocols.com>.

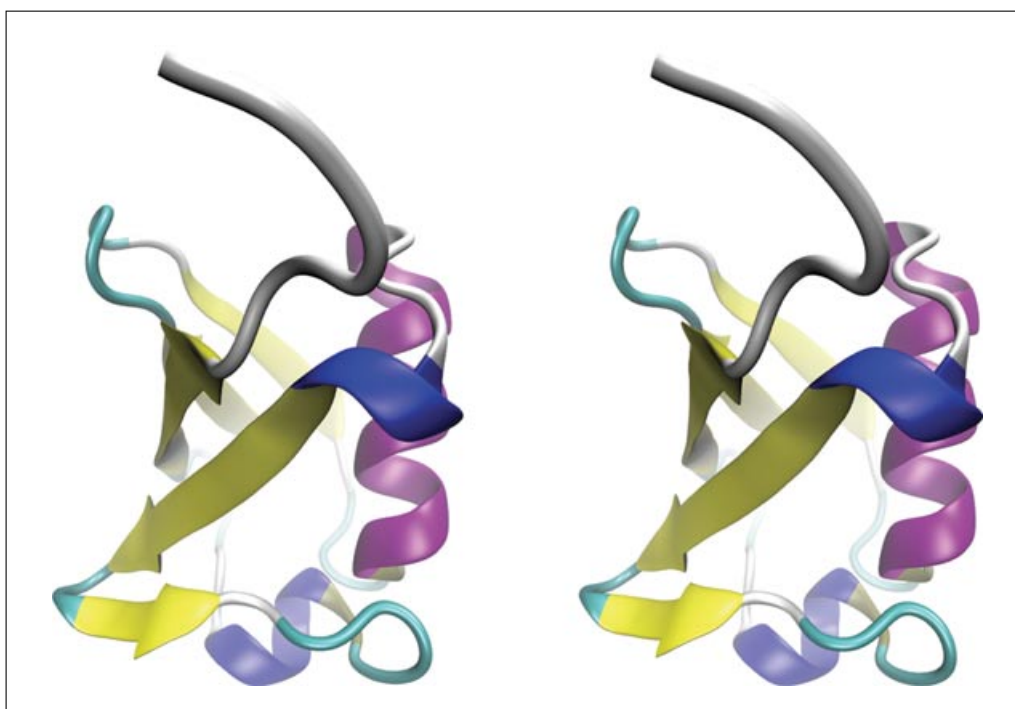


Figure 5.7.13 Stereo image of the ubiquitin protein. Shown here with Cue Mode = Linear, Cue Start = 1.5, and Cue End = 2.75. To view the stereo image, use the “wall-eyed” method: hold the page close to eyes, and shift the focus beyond the page until the two images overlap to form a three-dimensional object. If this is difficult, try scaling down the figure to a smaller size. This will make viewing easier. For the color version of this figure go to <http://www.currentprotocols.com>.

Rendering

By now, we have seen some techniques for producing nice views and representations of the molecule loaded in VMD. Now, we will explore the use of the VMD built-in snapshot feature and external rendering programs to produce high-quality images of your molecule. The “snapshot” renderer saves the on-screen image in the OpenGL window and is adequate for use in presentations, movies, and small figures. When one desires higher-quality images, renderers such as Tachyon and POV-Ray are better choices.

19. Hide or delete all previous representations, and create the four new representations listed in Table 5.7.7.

Table 5.7.7 Example Representations

Selection	Coloring method	Drawing style	Material
protein and not resid 72 to 76	Structure	NewCartoon	Opaque
protein and helix and name CA	ColorID→8	Surf	Material 12
resname GLY and not resid 72 to 76	ColorID→7	VDW	Opaque
resname LYS	ColorID→18	Licorice	Opaque

20. Once you have the scene set the way you like it in the OpenGL window, simply choose File → Render. . . in the VMD Main window. The File Render Controls window will appear on the screen.

21. The File Render Controls allows you to choose which renderer you want to use and the file name for your image. Select “snapshot” for the rendering method, type in a filename of your choice, and click Start Rendering.

22. If you are using a Mac or a Linux machine, an image-processing application might open automatically that shows you the molecule you have just rendered using Snapshot. If this is not the case, use any image-processing application to take a look at the image file. Close the application when you are done to continue using VMD.

The snapshot renderer saves exactly what is showing in your OpenGL display window—in fact, if another window overlaps the display window, it may distort the overlapped region of the image.

23. Try to render again using different rendering methods, particularly TachyonInternal and POV3 (see Fig. 5.7.14 for an example POV3 rendering). Compare the quality of the images created by different renderers.

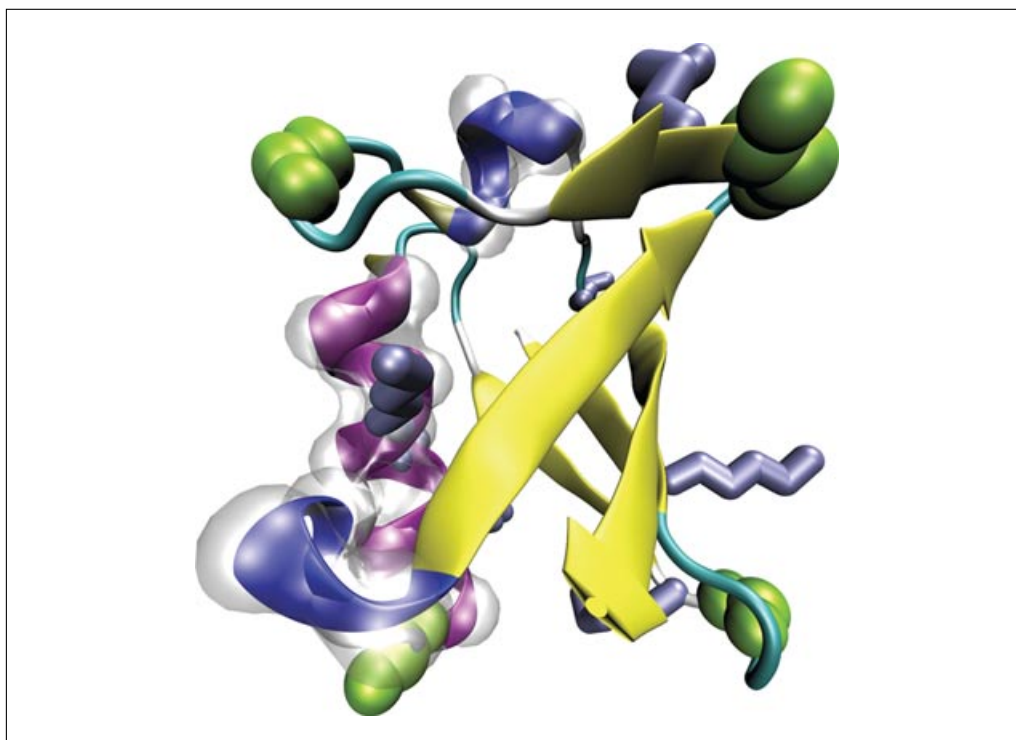


Figure 5.7.14 Example of a POV3 rendering. For the color version of this figure go to <http://www.currentprotocols.com>.

The other renderers (e.g., POV3 and Tachyon) reprocess everything, so it may not look exactly as it does in the OpenGL window. In particular, they do not “clip,” or hide, objects very near the camera. If you select Display → Display Settings... in the VMD Main window, you can set Near Clip to 0.01 to get a better idea of what will appear in your rendering.

24. Quit VMD.

WORKING WITH TRAJECTORIES AND MAKING MOVIES

Time-evolving coordinates of a system are called trajectories. They are most commonly obtained from simulations of molecular systems, but can also be generated by other means and for different purposes. Upon loading a trajectory into VMD, one can see a movie of how the system evolves in time and analyze various features throughout the trajectory. This section will introduce the basics of working with trajectory data in VMD. You will also learn how to analyze trajectory data in Basic Protocols 14, 15, and 16.

Necessary Resources

Hardware

Computer

Software

VMD, and a movie player program

Files

`ubiquitin.psf` and `pulling.dcd`, which can be downloaded from
<http://www.currentprotocols.com>

Working with Trajectories

Trajectory files are commonly binary files that contain several sets of coordinates for the system. Each set of coordinates corresponds to one frame in time. An example of a trajectory file is a DCD file generated by the molecular dynamics program NAMD (Phillips et al., 2005).

Load trajectories

Trajectory files do not contain information of the system contained in the protein structure files (PSF). Therefore, we first need to load the PSF file, and then add the trajectory data to this file.

1. Start a new VMD session. In the VMD Main window, select File → New Molecule. . . The Molecule File Browser window will appear on your screen.
2. Use the Browse... button to find the file `ubiquitin.psf`. When you select this file, you will be back in the Molecule File Browser window. Press the Load button to load the molecule.
3. In the Molecule File Browser window, make sure that `ubiquitin.psf` is selected in the “Load files for:” pull-down menu on top, and click on the Browse button. Browse for `pulling.dcd`.

Note the options available in the Molecule File Browser window: one can load trajectories starting and finishing at chosen frames, and adjust the stride between the loaded frames. Leave the default settings so that the whole trajectory is loaded.

4. Click on the Load button in the Molecule File Browser window.

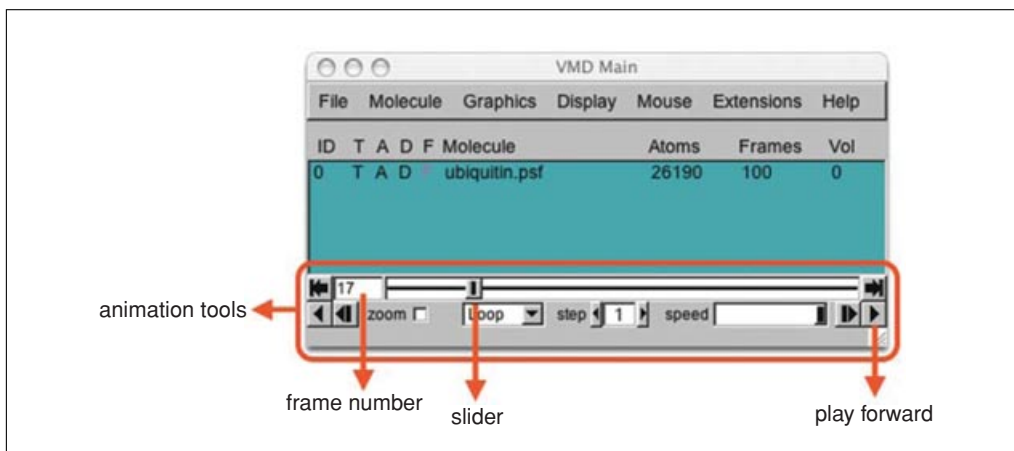


Figure 5.7.15 Animation tools in the VMD main menu. The tools allow one to go over frames of the trajectory (e.g., using the “slider”) and to play a movie of the trajectory in various modes (Once, Loop, or Rock) and at an adjustable speed.

You will be able to see the frames as they are loaded into the molecule in the OpenGL window. After the trajectory finishes loading, you will be looking at the last frame of your trajectory. To go to the beginning, use the animation tools at the lower part of the VMD Main menu (see Fig. 5.7.15).

5. Close the Molecule File Browser window.
6. For a convenient visualization of the protein, choose Graphics → Representations in the VMD Main menu. In the Selected Atoms text field, type **protein** and hit Enter on your keyboard; in the Drawing Method, select NewCartoon; in the Coloring Method, select Structure.

The trajectory you just loaded is a simulation of an AFM (Atomic Force Microscopy) experiment pulling on a single ubiquitin molecule, performed using the Steered Molecular Dynamics (SMD) method (Isralewitz et al., 2001). We are looking at the behavior of the protein as it unfolds while being pulled from one end, with the other end constrained to its original position. Each frame corresponds to 10 picoseconds of simulation time. Ubiquitin has many functions in the cell, and it is currently believed that some of these functions depend on the protein’s elastic properties, which can be probed in AFM pulling experiments. Such elastic properties are usually due to hydrogen bonding between residues in β strands of the protein molecules.

Using Main menu animation tools

You can now play the movie of the loaded trajectory back and forth, using the animation tools in Figure 5.7.15.

7. By dragging the slider (Fig. 5.7.15), one navigates through the trajectory. The buttons to the left and to the right from the slider panel allow one to jump to the end of the trajectory or go back to the beginning.
8. For example, create another representation for water in the Graphical Representations window: click on the Create Rep button; in the Selected Atoms field, type **water** and hit enter; for Drawing Method, choose Lines; for Coloring Method, select Name.

This representation of water shows the water droplet present in the simulation. Using the slider, observe the behavior of the water around the protein. The shape of the water droplet changes throughout the simulation, because water molecules follow the protein as it unfolds, driven by the interactions with the protein surface.

When playing animations, you can choose between three looping styles: Once, Loop, and Rock. You can also jump to a frame in the trajectory by entering the frame number in the window on the left of the “slider” panel.

Smoothing trajectories

9. For clarity, turn off the water representation by double-clicking on it in the Graphical Representations window.

As you might have noticed, when we play the animation, the protein movements are not very smooth due to thermal fluctuations (as the simulation is performed under the conditions that mimic a thermal bath). VMD can smooth the animation by averaging over a given number of frames.

10. In the Graphical Representations window, select your protein representation and click on the Trajectory tab. At the bottom, you should see the Trajectory Smoothing Window Size set to zero. As your animation is playing, increase this setting. Notice that the motion gets smoother and smoother as the size of the smoothing window is increased. Commonly used values for this setting are 1 to 5, depending on how smooth you want your trajectory to be.

Displaying multiple frames

We will now learn how to display many frames of the same trajectory at once.

11. In the Graphical Representations window, highlight your protein representation by clicking on it and press the Create Rep button. This creates an identical representation, but note that smoothing is set to zero. Hide the old protein representation.
12. Highlight the new protein representation and click the Trajectory tab. Above the smoothing control, notice the Draw Multiple Frames control. It is set to now by default, which is simply the current frame. Enter 0:10:99, which selects every tenth frame from the range 0 to 99.
13. Go back to the Draw style tab, and change the Coloring Method to Timestep. This will draw the beginning of the trajectory in red, the middle in white, and the end in blue.
14. We can also use smoothing to make the large-scale motion of the protein more apparent. Go back to the Trajectory tab, and set the smoothing window to 20. The result should look like Figure 5.7.16.

Updating selections

Now, we will see how to make VMD update the selection each frame.

15. Hide the current representation showing all frames, and display only the water representation by double-clicking on it. Change the text in the Selected Atoms field from `water` to `water and within 3 of protein` and hit enter. This will show all water atoms within 3 Å of the protein.
16. Play the trajectory.

As you can see, although the displayed water atoms may be near the protein for a little while, they soon wander off, and are still shown despite no longer meeting the selection criteria. The Update Selection Every Frame option in the Trajectory tab of the Graphical Representations window remedies this. If the option box is checked, the selection is updated every frame. See Figure 5.7.17.

17. Quit VMD.

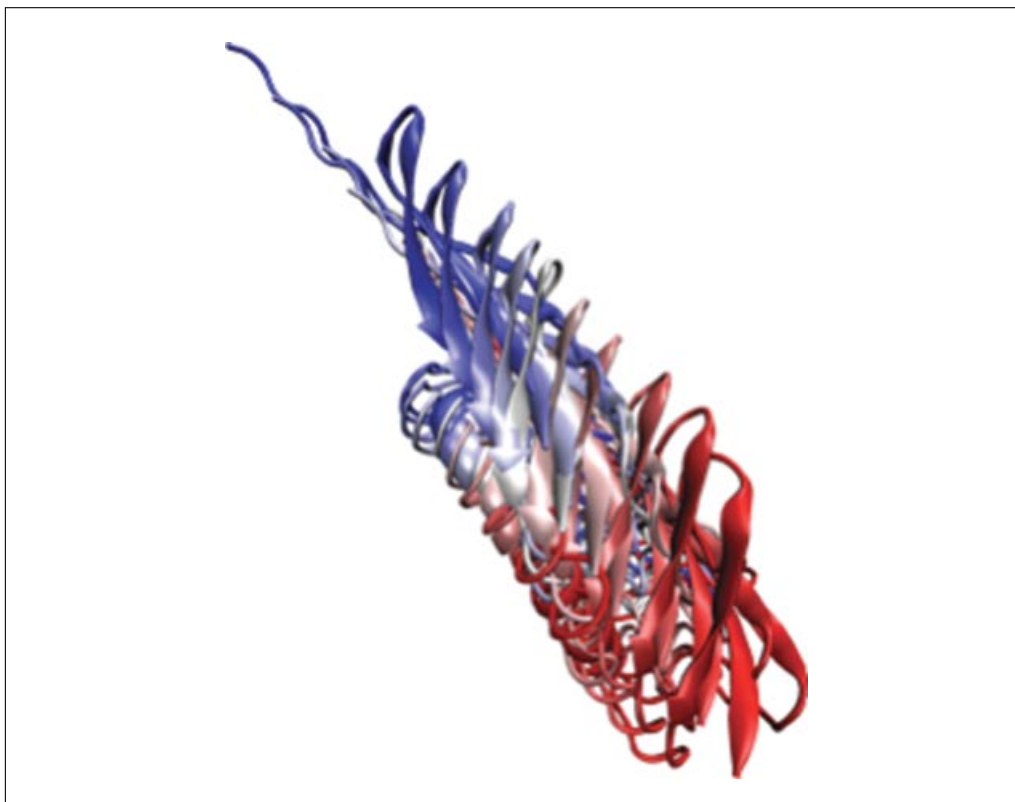


Figure 5.7.16 Image of every tenth frame shown at once, smoothed with a 20-frame window. For the color version of this figure go to <http://www.currentprotocols.com>.

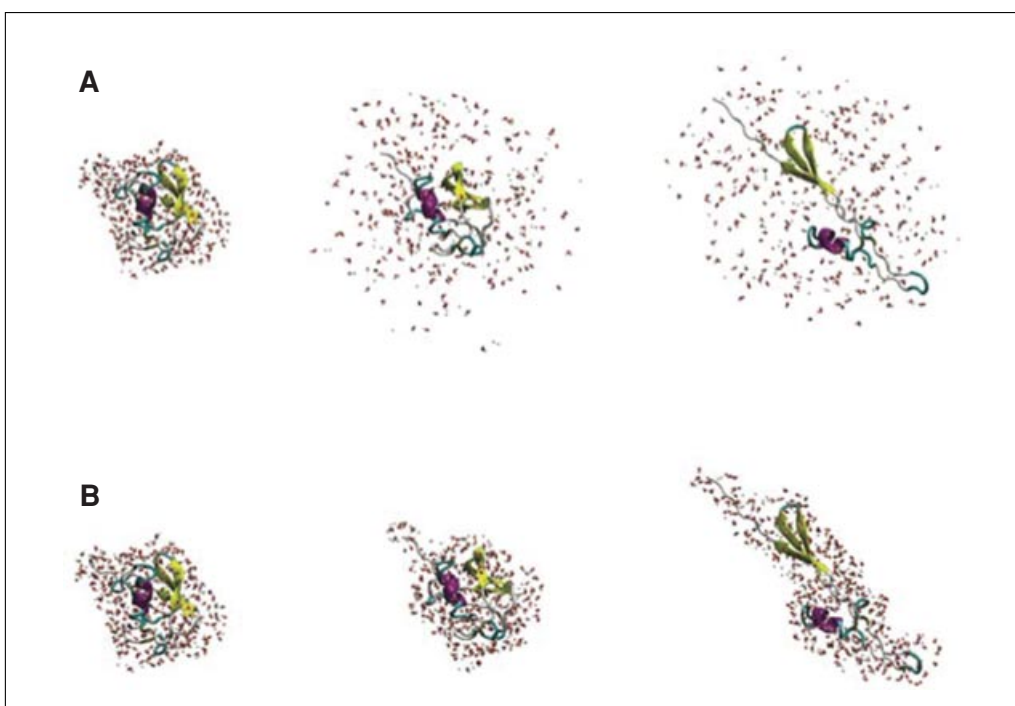


Figure 5.7.17 Water within 3 Å of the protein, shown for a selection that is not updated (**A**) and for the one that is updated (**B**) each frame. The snapshots shown are (from left to right) for frames 0, 17, and 99. For the color version of this figure go to <http://www.currentprotocols.com>.

The Basics of Movie Making in VMD

The following protocol describes how to make a movie in VMD.

1. Start a new VMD session. Repeat steps 1 to 5 of Basic Protocol 3 to load the ubiquitin trajectory into VMD and display the protein in a secondary structure representation.
2. To make movies, we will use the VMD Movie Maker plugin. In the VMD Main window, go to menu item Extension → Visualization → Movie Maker. The VMD Movie Generator window will appear.

Making single-frame movies

3. Click on the Movie Settings menu in the VMD Movie Generator window; take a look at the options.

You can see that in addition to a trajectory movie, Movie Maker can also make a movie by rotating the view point of a single frame. In the Renderer menu, one can choose the type of renderer for making the movie. While renderers other than Snapshot, e.g., Tachyon, generally provide more visually appealing images, they also take longer to render. The rendering time is also affected by the size of the OpenGL window, since it takes more computing time to render a larger image. We will first make a movie of just one frame of the trajectory. Here, we will use the default option, Snapshot. One can also choose the output file format for the movie in menu item Format.

4. Select the Rock and Roll option in the Movie Settings menu in the VMD Movie Generator window. Set the working directory to any convenient directory of your choice, give your movie a name, and click Make Movie.
5. Once rendering is finished, open and view the movie with your favorite application. This movie setting is good for showing one side of your system primarily.

If you cannot successfully make movies with VMD, it is possible that you are missing some software required for generating movies. All of the required softwares are freely available, and to find what software you need, please see the VMD Movie Plugin page at <http://www.ks.uiuc.edu/Research/vmd/plugins/vmdmovie/>.

Making trajectory movies

6. Now, we will make a movie of the trajectory. In the VMD Movie Generator window, select Movie Settings → Trajectory, give this one a different name, and click Make Movie.

Note that the length of the movie is automatically set 24 frames per second. For a trajectory, duration of the movie can be decreased, but cannot be increased.

7. Try out different options in the VMD Movie Generator window. Once you are done, quit VMD.

SCRIPTING IN VMD

VMD provides embedded scripting languages (Python and Tcl) for the purpose of user extensibility. In this section, we will discuss the basic features of the Tcl scripting interface in VMD. You will see that everything you can do in VMD interactively can also be done with Tcl commands and scripts. We will also demonstrate how the extensive list of Tcl text commands can help you investigate molecule properties and perform various types of analysis.

Necessary Resources

Hardware

Computer

VMD, and a text editor

Files

1ubq.pdb and beta.tcl, which can be downloaded from
<http://www.currentprotocols.com>

The Basics of Tcl Scripting

Tcl is a rich language that contains many features and commands, in addition to the typical conditional and looping expressions. Tk is an extension to Tcl that permits the writing of graphical user interfaces with windows and buttons, etc. More information and documentations about the Tcl/Tk language can be found at <http://www.tcl.tk/doc>. Let us start with the basic commands.

1. Start a new VMD session. In the VMD Main menu, select Extensions → Tk Console to open the VMD TkConsole window. You can now start entering Tcl/Tk commands here.
2. Try entering the following commands in the VMD TkConsole window. Remember to hit enter after each line and take a look at what you get after each input.

```
set x 10
puts 'the value of x is: $x'
set text 'some text'
puts 'the value of text is: $text'
```

As you can see, the Tcl `set` and `put` commands have the following syntax:

`set variable value` - sets the value of variable

`puts $variable` - prints out the value of variable

Also, `$variable` refers to the value of variable.

3. Try the `expr` command by entering the following lines in the VMD TkConsole window:

```
expr 3 - 8
set x 10
expr -3 * $x
```

The `expr` command performs mathematical operations:

`expr expression` - evaluates a mathematical expression.

4. Entering the following example in the VMD TkConsole window:

```
set result [expr -3 * $x]
puts $result
```

By using brackets, you can embed Tcl commands into others. A bracketed expression will automatically be substituted by the return value of the expression inside the brackets: `[expression]` - represents the result of the expression inside the brackets.

5. Let us calculate the values of $-3*x$ for integers x from 0 to 10 and output the results into a file named `myoutput.dat`.

```
set file [open 'myoutput.dat' w]
for {set x 0} {$x <= 10} {incr x} {
  puts $file [expr -3 * $x]
}
close $file
```

Here, you have tried the loop feature of Tcl. Tcl provides an iterated loop similar to the “for” loop in C. The `for` command in Tcl requires four arguments: an initialization, a test, an increment, and the block of code to evaluate. The syntax of the `for` command is:

```
for {initialization} {test} {increment} {commands}
```

Take a look at the output file `myoutput.dat`, either by a text editor of your choice, or the command `less` in a terminal window on a Mac or Linux Machine.

Working with a Molecule Using Tcl Text Commands

Anything that can be done in the VMD graphical interface can also be done with text commands. This allows scripts to be written that can automatically load molecules, create representations, analyze data, make movies, etc. Here, we will go through some simple examples of what can be done using the scripting interface in VMD.

Loading molecules with text commands

1. In the VMD TkConsole window, type the command `mol new lubq.pdb` and hit enter.

As you can see, this command performs the same function as described at the beginning of Basic Protocol 1, namely, loading a new molecule with file name `lubq.pdb`.

If you see the error message `Unable to load file 'lubq.pdb' using file type 'pdb'`, you might not be in the correct directory that contains the file `lubq.pdb`. You can use the standard Unix commands in the VMD TkConsole window to navigate to the correct directory.

When you open VMD, by default a vmd console window appears. The vmd console window tells you what’s going on within the VMD session that you are working on. Take a look at the vmd console window. It should tell you a molecule has been loaded, as well as some of its basic properties like number of atoms, bonds, residues etc. The Tcl commands that you enter in the VMD TkConsole window can also be entered in the vmd console window. If you are using a Mac, your vmd console window is the terminal window that shows up when you open VMD.

Working with specific parts of a molecule: the `atomselect` command

Many times, you might want to perform operations on only a specific part a molecule. For this purpose, VMD’s `atomselect` command is very useful. The `atomselect` command has the following syntax:

```
atomselect molid “selection command” - creates a new atom selection that includes all atoms described by “selection command”.
```

2. Type `set crystal [atomselect top 'all']` in the Tk Console window.

This command allows you to select a specific part of a molecule. The first argument to `atomselect` is the molecule ID (shown to the very left of the VMD Main window); the second argument is a textual atom selection like what you have been using to describe graphical representations in Basic Protocol 1. The selection returned by `atomselect` is itself a command you will learn to use.

This step creates a selection, `crystal`, that contains all the atoms in the molecule and assigns it to the variable `crystal`. Instead of a molecule ID (which is a number), we have used the shortcut `top` to refer to the top molecule. A top molecule means that it is the target for scripting commands. This concept is particularly important when multiple molecules are loaded at the same time (see Basic Protocol 9 for dealing with multiple molecules in VMD).

The result of `atomselect` is a function. Thus, `$crystal` is now a function that performs actions on the contents of the `'all'` selection.

Obtaining and changing molecule properties with text commands

After you have defined an atom selection, you have many commands that you can use to operate on it. For example, you can use commands to learn about the properties of your atom selection (number of atoms, coordinates, total charge, etc). You can also use commands to change its coordinates and other properties. See VMD User's Guide (<http://www.ks.uiuc.edu/Research/vmd/vmd-1.8.6/ug/>) for an extensive list of commands.

3. Type `$crystal num` in the Tk Console window.

Passing `num` to an atom selection returns the number of atoms in that selection. Check that this number matches the number of atoms for your molecule displayed in the VMD Main window.

4. We can also use commands to move our molecule on the screen. You can use these commands to change atom coordinates.

```
$crystal moveby {10 0 0}
$crystal move [transaxis x 40 degree]
```

Editing properties of selected atoms

5. Open the Graphical Representation window by selecting Graphics → Representations... in the VMD Main window. Type in `protein` as the atom selection; change its Coloring Method to Beta and its Drawing Method to VDW. Your molecule should now appear as a mostly red and blue assembly of spheres.

The “B” field of a PDB file typically stores the “temperature factor” for a crystal structure and is read into VMD’s “Beta” field. Since we are not currently interested in this information, we can use this field to store our own numerical values. VMD has a “Beta” coloring method, which colors atoms according to their β -factors. By replacing the Beta values for various atoms, you can control the color in which they are drawn. This is very useful when you want to show a property of the system that you have computed.

6. Return to the Tk Console window and type `$crystal set beta 0`.

This resets the “beta” field (which is displayed) to zero for all atoms. As you do this, you should observe that the atoms in your OpenGL window will suddenly change to a uniform color (since they all have the same beta values now).

You can obtain and set many atomic properties using atom selections, including segment, chain, residue, atom name, position (x, y and z), charge, mass, occupancy and radius, just to name a few.

7. In the Tk Console window, type `set sel [atomselect top 'hydrophobic']`.

This creates a selection, `sel`, that contains all the atoms in the hydrophobic residues.

8. Let us label all hydrophobic atoms by setting their beta values to 1: type `$sel set beta 1` in the Tk Console window. If the colors in the OpenGL Display do not get updated, go to the Graphical Representations window and click on the Apply button at the bottom.

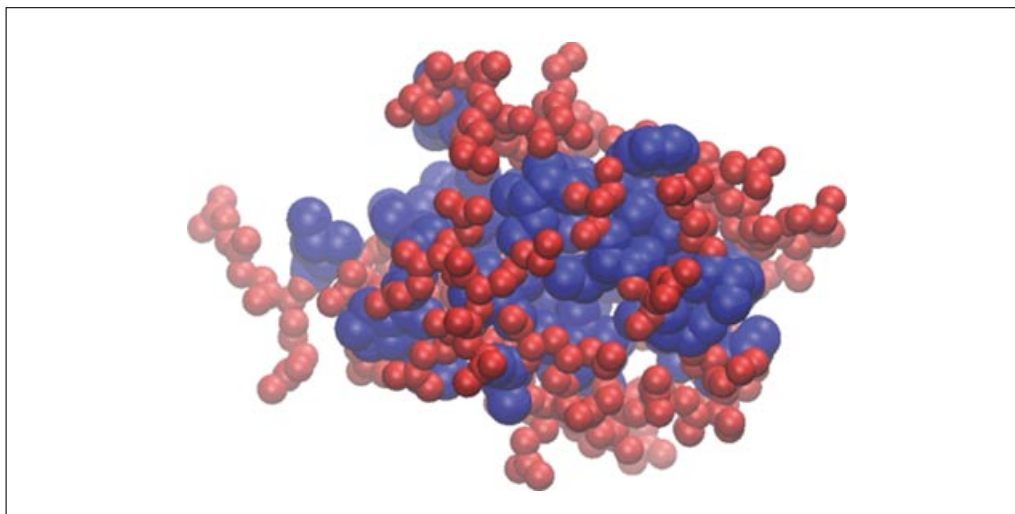


Figure 5.7.18 Ubiquitin in the VDW representation, colored according to the hydrophobicity of its residues. For the color version of this figure go to <http://www.currentprotocols.com>.

9. You will now change a physical property of the atoms to further illustrate the distribution of hydrophobic residues. In the Tk Console window type `$crystal set radius 1.0` to make all the atoms smaller and easier to see through, and then `$sel set radius 1.5` to make atoms in the hydrophobic residues larger. The radius field affects the way that some representations (e.g., VDW, CPK) are drawn.

You have now created a visual state that clearly distinguishes which parts of the protein are hydrophobic and which are hydrophilic. If you have followed the instructions correctly, your protein should resemble Figure 5.7.18.

Many times in studies of proteins, it is important to identify the locations of the hydrophobic residues, as they often have a functional implication. The method you have just learned is useful in this task. For example, you can easily see that in ubiquitin, the hydrophobic residues are almost exclusively contained in the inner core of the protein. This is a typical feature for small water-soluble proteins. As the protein folds, the hydrophilic residues will have a tendency to stay at the water interface, while the hydrophobic residues are pushed together. This helps the protein achieve proper folding and increases its stability.

The get command

Atom selections are useful not only for setting atomic data, but also for getting atomic information. For example, if you wish to communicate which residues are hydrophobic, all you need to do is to create a hydrophobic selection and use the `get` command.

10. Try to use the `get` command with your `sel` atom selection to obtain the names of hydrophobic residues:

```
$sel get resname
```

But there is a problem; each residue contains many atoms, resulting in multiple repeated entries. One way to circumvent this is to pick only the α -carbons in the selection.

11. Type the following in the Tk Console window (note, name `CA` = α -carbons):

```
set sel [atomselect top 'hydrophobic and name CA']
$sel get resname
```

This should give you the list of hydrophobic residues.

12. You can also get multiple properties simultaneously. Try the following:

```
$sel get resid  
$sel get {resname resid}  
$sel get {x y z}
```

If you want to obtain some of the structural properties, e.g., the geometric center or the size of a selection, the command `measure` can do the job easily.

13. Let us try using `measure` with the `sel` selection:

```
measure center $sel  
measure minmax $sel
```

The first command above returns the geometric center of atoms in `sel`. And the second command returns two vectors, the first containing the minimum x, y, and z coordinates of all atoms in `sel`, and the second containing the corresponding maxima.

Once you are done with a selection, it is always a good idea to delete it to save memory:

```
$sel delete
```

Sourcing Scripts

When performing a task that requires many lines of commands, instead of typing each line in the Tk Console window, it is usually more convenient to write all the lines into a script file and load it into VMD. This is very easy to do. Just use any text editor to write your script file, and in a VMD session, use the command `source filename` to execute the file. You should have downloaded a simple script, `beta.tcl`, with this unit. We will execute it in VMD as an example. The script `beta.tcl` sets the colors of residues LYS and GLY to a different color from the rest of the protein by assigning them a different beta value, a trick you have already learned in Basic Protocol 6, steps 5 to 9.

In the Tk Console window, type `source beta.tcl` and observe the color change. You should see that the protein is mostly a collection of red spheres, with some residues shown in blue. The blue residues are the LYS and GLY residues in the ubiquitin. Take a quick look at the script `beta.tcl`. Using any text editor of your choice, open the file `beta.tcl`. There are six lines in this file, and each line represents a Tcl command line that you have used before. Close the text editor when you are done.

The `.vmd` file you saved in Basic Protocol 1, step 47, is actually a series of commands. You are encouraged to take a look at that file using a text editor. Hopefully, by the end of this section, you'll understand many of those commands. In fact, you can execute the file in the Tk Console the same way as you execute other script files, i.e., by typing `source myfirststate.vmd` in the Tk Console window.

Many times when you write a script you might want to look up the command for an interactive VMD feature. You can either find it in the VMD User's Guide (<http://www.ks.uiuc.edu/Research/vmd/vmd-1.8.6/ug/>) or conveniently use the console command. Try typing `logfile console` in your Console window. This creates a logfile for all your actions in VMD and writes them in the Console window as command lines. If you execute those command lines, you can repeat the exact same actions you have performed interactively. To turn off logfile, type `logfile off`.

BASIC PROTOCOL 7

Modeling Structure from Sequence

5.7.27

Drawing Shapes Using VMD Text Commands

VMD offers a way to display user-defined objects built from graphics primitives such as points, lines, cylinders, cones, spheres, triangles, and text. The command that can realize those functions is `graphics`, the syntax of which is `graphics molid command`, where `molid` is a valid molecule ID and `command` is one of the commands shown below. Let us try drawing some shapes with the following examples.

1. Hide all representations in the Graphics Representations window.
2. Let us draw a point. Type the following command in your Tk Console window:

```
graphics top point {0 0 10}
```

Somewhere in your OpenGL window, there should be a small dot.

3. Let us draw a line. Type the following command in your Console window (note the “\” in command line means the next line is a continuation of the previous line, hence do not actually type “\” when you enter the following command, and do not start a new line):

```
graphics top line {-10 0 0} {0 0 0} width 5 style\  
solid
```

This will give you a solid line.

4. You can also draw a dashed line:

```
graphics top line {10 0 0} {0 0 0} width 5 style\  
dashed
```

All the objects so far are all drawn in blue. You can change the color of the next graphics object by using the command `graphics top color colorid`. The `colorid` for each color can be found in Graphics → Colors... menu in VMD Main window. For example, the color for orange is “3.”

5. Type `graphics top color 3` in the Tk Console window and the next object you draw will appear in orange.
6. Try the following commands to draw more shapes:

```
graphics top cylinder {5 0 0} {15 0 10} radius 10\  
resolution 60 filled no  
graphics top cylinder {0 0 0} {-5 0 10} radius 5\  
resolution 60 filled yes  
graphics top cone {40 0 0} {40 0 10} radius 10\  
resolution 60  
graphics top triangle {80 0 0} {85 0 10} {90 0 0}  
graphics top text {40 0 20} ''my drawing objects''
```

7. In your OpenGL window, there are a lot of objects now. To find the list of objects you’ve drawn, use the command `graphics top list`. You’ll get a list of numbers, standing for the ID of each object.
8. The detailed information about each object can be obtained by typing `graphics top info ID`. For example, type `graphics top info 0` to see the information on the point you drew.

9. You can also delete some of the unwanted objects using the command `graphics top delete ID`.

Using these basic shape-drawing commands, you can create geometrical objects, as well as text, to be displayed in your OpenGL window. When you render an image (as discussed in Basic Protocol 2, steps 19 to 23), these objects will be included in the resulting image file. You can hence use geometric objects and texts to point or label interesting features in your molecule, for example, an arrow (a combination of a cylinder and a cone) can be drawn this way to point at a region of interest of your molecule

10. Quit VMD.

WORKING WITH MULTIPLE MOLECULES

In this section, you will learn to work with multiple molecules within one VMD session. We will use the water transporting channel protein, aquaporin, as an example.

Necessary Resources

Hardware

Computer

Software

VMD

Files

`1fqy.pdb` and `1rc2.pdb`, which can be downloaded at <http://www.currentprotocols.com>

Molecule List Browser

Aquaporins are membrane channel proteins found in a wide range of species, from bacteria to plants to human. They facilitate water transport across the cell membrane, and play an important role in the control of cell volume and transcellular water traffic. Many aquaporin protein structures are available in the Protein Data Bank, including a human aquaporin (PDB code 1FQY; Murata et al., 2000) and an *E. coli* aquaporin (PDB code 1RC2; Savage et al., 2003). To practice dealing with multiple proteins in VMD, let us load both aquaporin structures.

Loading multiple molecules

1. Start a new VMD session. In the VMD Main window, choose File → New Molecule. . . The Molecule File Browser window should appear on your screen.
2. Use the Browse. . . button to find the file `1fqy.pdb`. When you select the file, you will be back in the Molecule File Browser window. Press the Load button to load the molecule. The coordinate file of human aquaporin AQP1 should now be loaded and can be seen in the OpenGL window.
3. In the Molecule File Browser, make sure you choose New Molecule in the Load files for: pull-down menu on the top. Use the Browse. . . button to find the file `1rc2.pdb` and press Load. Close the Molecule File Browser window.

You have just loaded two molecules. Any number of molecules can be loaded and displayed in VMD simultaneously by repeating the previous step. VMD can load as many molecules as the memory of your computer allows.

Take a look at your VMD Main window, which should look like Figure 5.7.19. Within the VMD Main menu you can find the Molecule List Browser (circled in Fig. 5.7.19), which shows the global status of the loaded molecules. The Molecule List Browser displays

BASIC PROTOCOL 9

Modeling Structure from Sequence

5.7.29

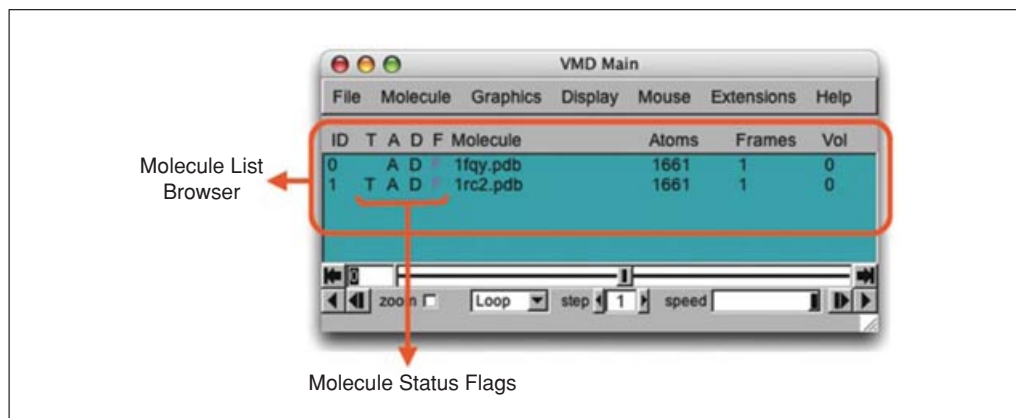


Figure 5.7.19 The Molecule List Browser.

information about each molecule, including Molecule ID (ID), the four Molecule Status Flags (T, A, D, and F, which stand for Top, Active, Drawn, and Fixed), name of the molecule (Molecule), number of atoms in the molecule (Atoms), number of frames loaded in the molecule (Frames), and the volumetric data loaded (Vol). Let us first start with the Molecule column. By default, the Molecule column displays file names of the molecules loaded in VMD, but you can change the molecule names to recognize them more easily.

Changing molecule names

4. In the VMD Main menu, double-click on 1fgy.pdb in the Molecule column. A window will pop up with the message Enter a new name for molecule 0:. Type in human aquaporin, and click OK (or press enter). In the VMD Main menu, the first molecule now has the name human aquaporin.
5. Repeat the previous step for the *E. coli* aquaporin by double-clicking the 1rc2.pdb molecule name, and changing it to *E. coli* aquaporin in the pop-up window.

Drawing different representations for different molecules

Before we continue exploring other features in the Molecule List Browser, take a look at your OpenGL Display window. You have two aquaporin structures, but since they are both shown in the same default representation, it is difficult to distinguish them. To tell them apart, you can assign them different representations.

6. Open the Graphical Representations window via Graphics → Representations... from the VMD Main menu. Make sure “0:human aquaporin” is selected in the Selected Molecule pull-down menu on top. Select NewCartoon for Drawing Method, and ColorID → 1 red for Coloring Method.
7. In the Graphical Representations window, select “1:*E. coli* aquaporin” in the Selected Molecule pull-down menu on top. Select NewCartoon for Drawing Method, and ColorID → 4 yellow for Coloring Method. Close the Graphical Representations window.

Now, your OpenGL Display window should show a human aquaporin colored in red and an *E. coli* aquaporin colored in yellow.

Molecule status flags

In your OpenGL Display window, try moving the aquaporins around with your mouse in different mouse modes (rotating, scaling, and translating). You can see that both aquaporins move together. You can fix any molecule by double-clicking the “F” (fixed) flag in the Molecule List Browser on the left of the molecule name.

8. In the Molecule List Browser, double-click on the “F” flag on the left of “human aquaporin” to fix the human aquaporin molecule. Return to the OpenGL Display window and toggle your mouse around. You can see that only the yellow *E. coli* aquaporin moves. Double-click on the “F” flag for human aquaporin again to release it.

One thing to notice about the “F” flag is that, although it may seem that one molecule has been moved relative to another when one of the molecules is fixed, the difference is only apparent. The internal coordinates of molecules are not changed by the rotation, translation, and scaling motions. To change the coordinates of atoms in a molecule you need to use the text command interface (discussed in Basic Protocol 6, step 4), or by using the atom move picking modes (by choosing Mouse→Move in the VMD Main menu).

Other features in the Molecule List Browser include the Molecule ID (ID), Top (T), Active (A), and Drawn (D). Molecule ID is a number (starting from 0) assigned to each molecule when it is loaded into VMD, and permits VMD to recognize each molecule internally. You also refer to molecules by their Molecule IDs in the text command interface. Top flag (T) indicates the default molecule in VMD operations, for example when resetting the VMD OpenGL view and when playing molecule trajectories. There can be only one top molecule at a time. Active flag (A) indicates if the trajectory of the given molecule is updated when using animation tools described in Basic Protocol 3.

Finally, Drawn flag (D) indicates if the given molecule is displayed in the OpenGL window. Let us try out the Top and Drawn flags.

9. Make sure no molecule is fixed. By default, the last molecule loaded in the VMD is the top molecule, so you can check and see that there is a “T” displayed for the *E. coli* aquaporin in the VMD Main menu.
10. Reset the view by pressing the “=” key on the keyboard while keeping the OpenGL Display window active. Note that the yellow *E. coli* aquaporin is now placed in the center of the OpenGL Display window.
11. Switch the top molecule by double-clicking on the empty “T” flag for the human aquaporin molecule in the VMD Main menu. A “T” should appear for the human aquaporin, while the “T” for *E. coli* disappears. Go to the OpenGL Display window and reset the view again. You can see that this time the red human aquaporin is placed in the center of the OpenGL Display window.
12. In the VMD Main menu, try hiding a molecule by double-clicking on its “D” flag. You can display the molecule again by double-clicking its “D” flag again.

Aligning Molecules with the **measure fit** Command

When you look at your OpenGL Display window, you can see that the two aquaporins are very similar in structure. But it is difficult to detect their slight structural differences as the two proteins are placed apart. We will now try out a very useful Tcl command **measure fit** to align two molecules.

Open the VMD TkConsole window by choosing Extension → TkConsole from the VMD Main menu, and input the following commands:

```
set sel0 [atomselect 0 all]
set sel1 [atomselect 1 all]
set M [measure fit $sel0 $sel1]
$sel0 move $M
```

measure fit selection1 selection2 – measures the transformation matrix that best aligns the coordinates of **selection1** with the coordinates of **selection2**.

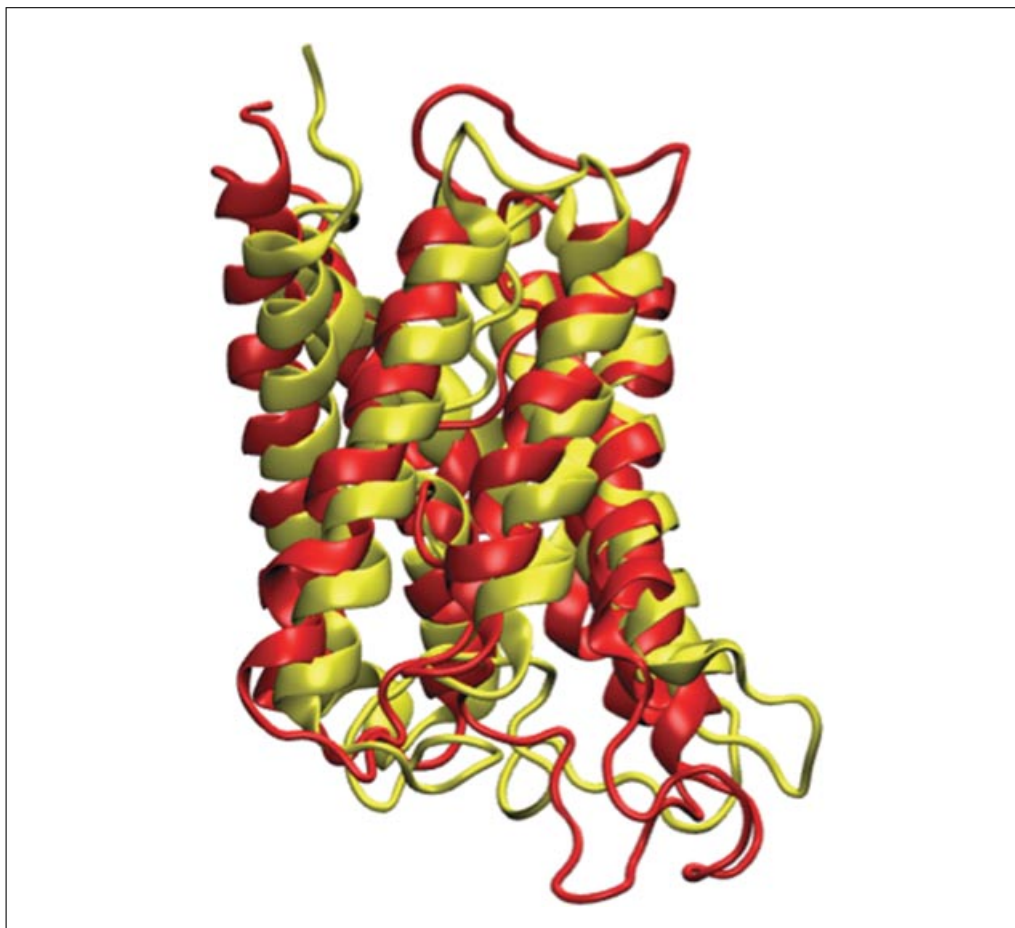


Figure 5.7.20 Result of the alignment between the two aquaporins using the `measure fit` command. For the color version of this figure go to <http://www.currentprotocols.com>.

As soon as you enter the last command line, you can see that the two aquaporins are now overlapping (Fig. 5.7.20). The α -helical regions of the aquaporins agree very well, with bigger deviations in the loop regions. Note that the `measure fit` command can only work if two molecules have the same number of atoms. In this case, it is a pure coincidence that the human aquaporin and *E. coli* aquaporin PDB files have the same number of atoms. The `measure fit` command is hence most useful in aligning the same protein in different conformations or different frames of a molecular dynamics simulation trajectory. Generally, to compare the structures of different proteins, one needs to use a different method. A good tool is the VMD MultiSeq plugin, which we will discuss in the following section.

COMPARING PROTEIN STRUCTURES AND SEQUENCES WITH THE MultiSeq PLUGIN

MultiSeq (Roberts et al., 2006) is a bioinformatics analysis environment developed in the Luthey-Schulten Group at the University of Illinois in Urbana-Champaign. MultiSeq allows users to organize, display, and analyze both sequence and structure data for proteins and nucleic acids, and has been incorporated in VMD as a plugin tool starting with VMD version 1.8.5 (MultiSeq homepage: <http://www.scs.uiuc.edu/~schulten/multiseq>). In this section, you will learn how to compare protein structures and sequences with the VMD MultiSeq plugin. We will again use the water transporting channel protein, aquaporin, as an example.

Necessary Resources

Hardware

Computer

Software

VMD, and a text editor

Files

1fqy.pdb, 1rc2.pdb, 1lda.pdb, 1j4n.pdb, and spinach_aqp.fasta, which can be downloaded at <http://www.currentprotocols.com>

Structure Alignment with MultiSeq

Very often comparing structures of different proteins reveals important information. For example, proteins with similar functions tend to exhibit similar structural features. MultiSeq structure alignment is useful for this reason. We will compare the structures of four aquaporin proteins listed in Table 5.7.8.

Loading aquaporin structures

1. Start a new VMD session. Open the Molecule File Browser window by choosing the File → New Molecule... menu item in the VMD Main window. In the Molecule File Browser window, use the Browse... button to find and select the file 1fqy.pdb. Press Load to load the molecule.
2. Load the remaining aquaporins, 1rc2, 1lda, and 1j4n. Make sure that each pdb file is loaded into a new molecule. Close the Molecule File Browser window when you have loaded all four molecules. Your VMD Main menu should look like Figure 5.7.21 when all four aquaporins are loaded.

Aligning the molecules

3. Within the VMD main window, choose the Extension menu and select Analysis → MultiSeq.

The MultiSeq window (with window name untitled.multiseq showing at the top) should now be open. You may be asked to update some databases in a pop-up window if this is the first time you use MultiSeq. If this is the case, simply click Yes and wait for MultiSeq to finish downloading. When MultiSeq starts, your MultiSeq window should display a list of the four aquaporin protein structures and a list of two nonprotein structures. The nonprotein structures are detergent molecules used in crystallizing the aquaporin proteins, and will not be needed for structure or sequence alignment. You can tell MultiSeq to discard molecules you are not interested in.

4. In the MultiSeq window, select the 1lda_X detergent molecule by clicking on it. This will highlight the entire row of 1lda_X. Remove it from MultiSeq by pressing

Table 5.7.8 The Four Aquaporins Used in this Section

PDB code	Description	Reference
1fqy	Human AQP1	Murata et al. (2000)
1rc2	<i>E. coli</i> AqpZ	Savage et al. (2003)
1lda	<i>E. coli</i> Glycerol Facilitator (GlpF)	Tajkhorshid et al. (2002)
1j4n	Bovine AQP1	Sui et al. (2001)

BASIC PROTOCOL 11

Modeling Structure from Sequence

5.7.33

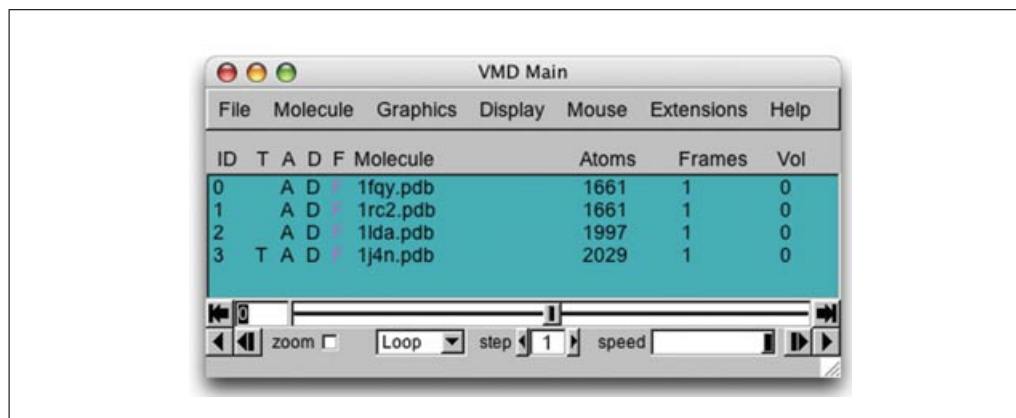


Figure 5.7.21 VMD Main menu after loading the four aquaporins.

the delete or Backspace key on your keyboard. Do the same to remove the 1j4n_X detergent molecule.

MultiSeq uses the program STAMP (Russell and Barton, 1992) to align protein molecules. STAMP (Structural Alignment of Multiple Proteins) is a tool for aligning protein sequences based on three-dimensional structures. Its algorithm minimizes the C_{α} distance between aligned residues of each molecule by applying globally optimal rigid-body rotations and translations. Note that you can only perform alignments on molecules that are structurally similar; if you try to align proteins that have no common structures, STAMP will fail.

5. In the MultiSeq window, select Tool → Stamp Structural Alignment. This will open the Stamp Alignment Options window.
6. In the Stamp Alignment Options window, choose Align the following: All Structures and go to the bottom of the menu and press OK.

The molecules have been aligned. You can see the alignment both in the OpenGL window and in the MultiSeq window (Fig. 5.7.22). Your alignment in OpenGL window will not immediately resemble Figure 5.7.22. When MultiSeq completes an alignment, it creates a new representation for all the aligned proteins in the NewCartoon representation with the same default coloring method and hides all other representations created previously. Let us give different colors to different aquaporins to distinguish them.

7. Open your Graphical Representations window, and you should see two representations for each molecule, the first one created when VMD loaded the molecule (which is now hidden), and the second one created automatically by MultiSeq. Select “0:1fqy.pdb” in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Change the color for this representation by selecting ColorID → 1 red for Coloring Method.
8. In the Graphical Representations window, select “1:1rc2.pdb” in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 4 yellow for Coloring Method.
9. In the Graphical Representations window, select “2:1lda.pdb” in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 11 purple for Coloring Method.
10. In the Graphical Representations window, select “3:1j4n.pdb” in the Selected Molecule pull-down menu on top and highlight the bottom representation by clicking on it. Select ColorID → 12 lime for Coloring Method. Close the Graphical Representations window.

Now your OpenGL window should look similar to Figure 5.7.22, and you can see that the alignment was pretty good as the four aquaporin structures are very similar.

You can get more information about the alignment in the MultiSeq window by highlighting the molecules you wish to compare.

11. In the MultiSeq window, highlight 1fqy by clicking on it.
12. To highlight another molecule without unhighlighting 1fqy, you need to Ctrl-click (or command-click on a Mac) on that molecule. Highlight 1rc2 by clicking on it while holding down the Ctrl key on the keyboard (or the command key on a Mac).

When both 1fqy and 1rc2 are highlighted, you should see at the lower left corner in the MultiSeq window a line of text: $Q_H:0.6442$, $RMSD:2.3043$, Percent Identity:30.28. Note that the values you obtain might be a little different depending on if your MultiSeq database is updated, but they should be close to the ones given here.

The Q_H value is a metric for structural homology. It is an adaptation of the Q value that measures structural conservation (Eastwood et al., 2001). $Q=1$ implies that structures are identical. When Q has a low score (0.1 to 0.3), structures are not aligned well, i.e., only a small fraction of C_α atoms superimpose. Along with RMSD and Percent Identity, these numbers tell you that the 1fqy and 1rc2 structures are pretty well aligned. You can repeat the previous step to compare the alignment of other molecules. To unselect a highlighted molecule, Ctrl-click on it again (or command-click on a Mac).

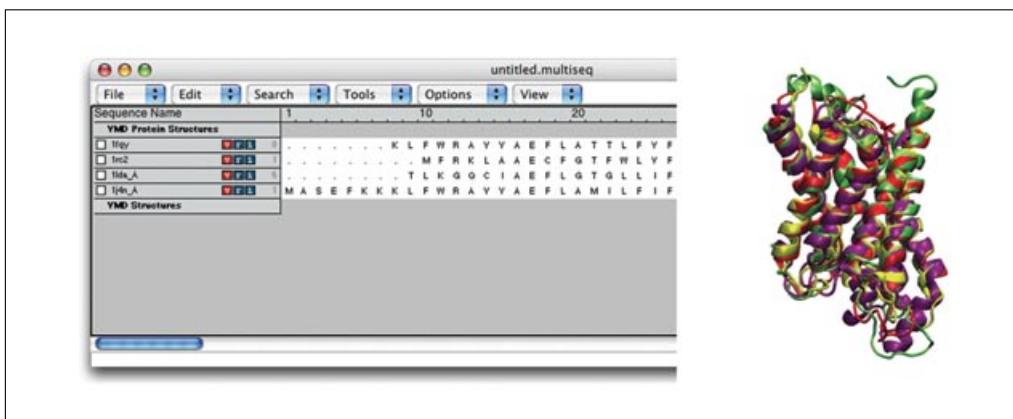


Figure 5.7.22 The four aquaporins aligned according to their structural similarity. For the color version of this figure go to <http://www.currentprotocols.com>.

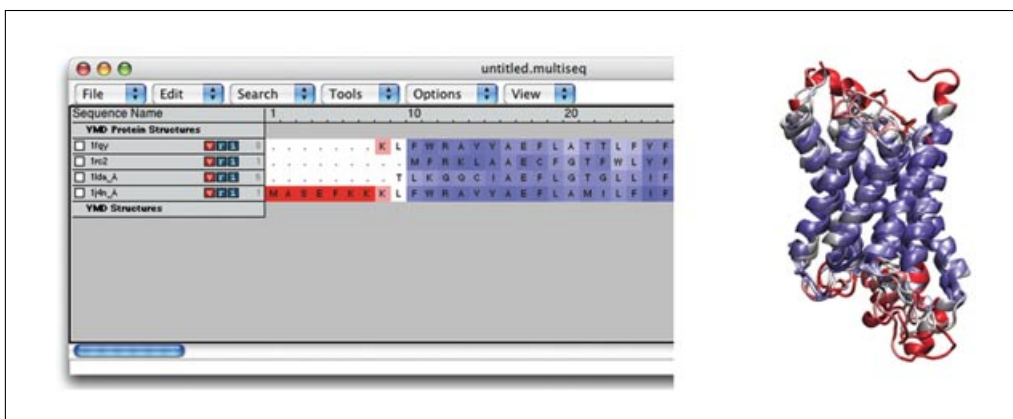


Figure 5.7.23 Result of a structural alignment of the four aquaporins, colored by Q_{res} . For the color version of this figure go to <http://www.currentprotocols.com>.

Coloring molecules according to structural identity

You can also color the molecules according to the value of Q per residue (Q_{res}) obtained in the alignment. Q_{res} is the contribution from each residue to the overall Q value of aligned structures.

13. In the MultiSeq window, choose View → Coloring → Qres.

Look at the OpenGL window to see the impact this selection has made on the coloring of the aligned molecules (Fig. 5.7.23). Blue areas indicate that the molecules are structurally conserved at those points, red areas indicates that there is no correspondence in structure at those points. As you can see, the α -helices that form the pore are well conserved structurally among the four aquaporins, while there are more structural differences in the less functionally relevant loops.

Sequence Alignment with MultiSeq

Besides revealing structural similarities, MultiSeq also allows comparison of proteins based on their sequences. Sequence alignment is often used to identify conserved residues among similar proteins, as such residues are likely of functional importance.

Aligning and coloring molecules by degree of conservation

1. In the MultiSeq window, select Tools → ClustalW Sequence Alignment.
2. In the ClustalW Alignment Options window, make sure the Align All Sequences option is checked, and go to the bottom of the window and select OK. Now the four aquaporins have been aligned according to their sequence using the ClustalW tool (Thompson et al., 1994).
3. Let us color the aligned molecules by their sequence similarity. In the MultiSeq window, choose View → Coloring → Sequence identity.

Now, each amino acid is colored according to the degree of conservation within the alignment: blue means highly conserved, red means low or no conservation. Your MultiSeq window and OpenGL window should resemble Figure 5.7.24.

You have now aligned the four aquaporins according to their sequence and identified the conserved residues, found mainly inside the pore (Fig. 5.7.25). Since aquaporin facilitates water transport across the membrane, these conserved residues are most likely the ones that carry out this function.

Importing FASTA files for sequence alignment

Many times the structure of a protein might not be available, but its sequence is. You can analyze a protein in MultiSeq without its structure by loading its sequence information



Figure 5.7.24 Result of a sequence alignment of the four aquaporins, colored by sequence identity. For the color version of this figure go to <http://www.currentprotocols.com>.

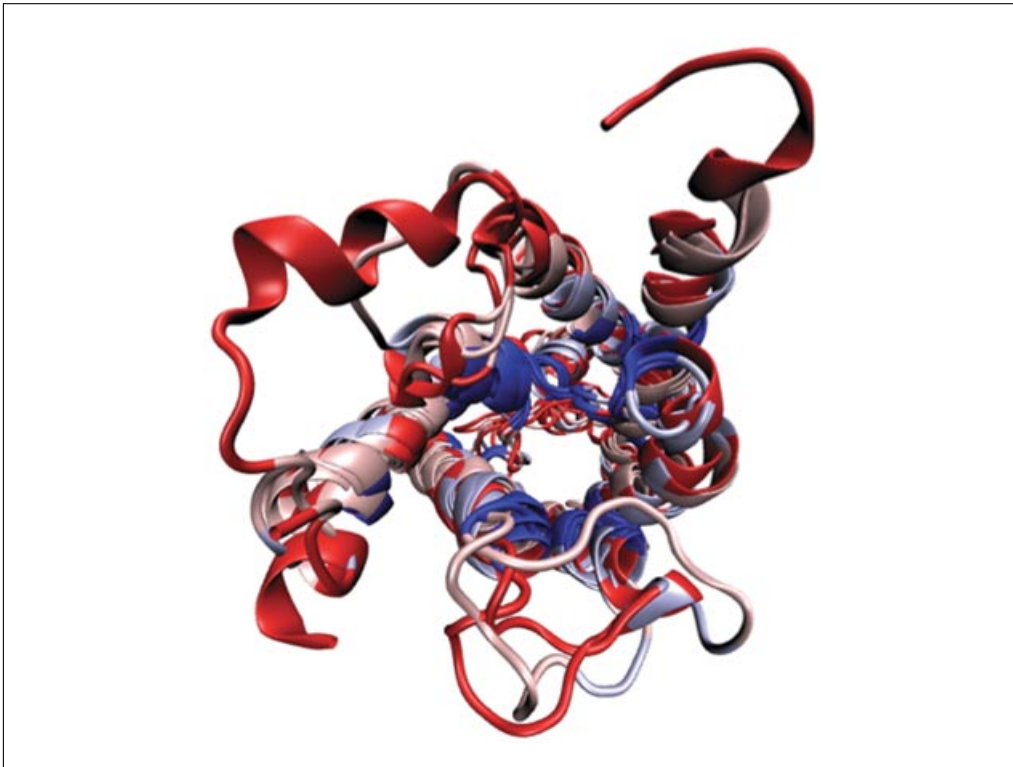


Figure 5.7.25 Top view of the aligned aquaporins colored by sequence conservation. The conserved residues locate mostly inside the aquaporin pore. For the color version of this figure go to <http://www.currentprotocols.com>.

in the FASTA file format. If you do not have the FASTA file of a protein but you have its sequence, you can create a FASTA file easily with any text editor of your choice.

4. Find the provided FASTA sequence file `spinach_aqp.fasta` and open it with a text editor.

A FASTA file contains a header that starts with ">" followed by the name of the protein. In the next line is the protein sequence in a one-letter amino acid code. You can create FASTA files similarly in this format. When you create a FASTA file, remember to save it in plain text, and use .fasta as the file extension.

5. Close the text editor when you finish examining `spinach_aqp.fasta`.
6. In the MultiSeq window, select File → Import Data... Select From File in the Import Data window, and press the top Browse button on top to select the file `spinach_aqp.fasta`. Press OK on the bottom of the Import Data window.

You have now loaded the sequence of a spinach aquaporin into MultiSeq. You can now perform sequence alignment on the spinach aquaporin protein with other loaded aquaporin molecules. Let us try a sequence alignment between a spinach and a human aquaporin.

7. Click on the checkbox on the left of `spinach_aqp`, and click on the checkbox on the left of `1fqy.pdb`.
8. Open the ClustalW Alignment Options window by selecting Tools → ClustalW Sequence Alignment. Under the Multiple Alignment options on the top, check Align Marked Sequences. Go to the bottom of the window and select OK.

The sequence of spinach aquaporin is now aligned with the sequence of human aquaporin, and you can check how good the alignment is by obtaining its Q_H and Sequence Identity values. If you feel that the two molecules are listed too far apart in the MultiSeq window, you can move the molecules by dragging them with your mouse. Also, as you might have noticed, in MultiSeq molecules can be Marked by checking their checkboxes. They can

also be Selected by highlighting them. You can align only the molecules of your choice by selecting *Align Marked Sequences* or *Align Selected Sequences*, depending if you have marked or highlighted your molecules. This option is available for both structural alignment and sequence alignment.

The structure of spinach aquaporin is actually available (Törnroth-Horsefield et al., 2006), but now that you have learned how to import FASTA sequence data, you can compare the sequences of proteins even if their structures are not resolved yet experimentally.

- When you finish comparing the sequence of spinach aquaporin with other aquaporins, delete it by clicking on `spinach_aqp` and press delete or Backspace on your keyboard.

BASIC PROTOCOL 13

Creating a Phylogenetic Tree with MultiSeq

The Phylogenetic Tree feature in MultiSeq elucidates the structure-based and/or sequence-based relationships between different proteins. Structure-based phylogenetic trees can be constructed according to the RMSD or Q values between the molecules after alignment; sequence-based phylogenetic trees can be constructed according to the percent identity or ClustalW values (Thompson et al., 1994).

- Align the structures again by going to the MultiSeq window and selecting Tools → Stamp Structural Alignment.
- In the Stamp Structural Alignment window, select All Structures, and keep the default values for the rest of the parameters. Press the OK button to align the structures.
- In the MultiSeq program window, choose Tools → Phylogenetic Tree. The Phylogenetic tree window will open.
- Select Structural tree using Q_H , and press the OK button.

*A phylogenetic tree based on the Q_H values should be calculated and drawn as shown in Figure 5.7.26A. Here you can see the relationship between the four aquaporins, e.g., how the *E. coli* AqpZ (1r2c) is related to human AQP1 (1fgy).*

- You can also construct the phylogenetic tree of the four aquaporins based on their sequence information. Close the Tree Viewer window.
- You need to perform the sequence alignment again for the four aquaporin proteins. In your MultiSeq window, choose Tools → ClustalW Sequence Alignment, and make sure the Align All Sequences option is checked, and press OK.
- In the MultiSeq program window, choose Tools → Phylogenetic Tree to open the Phylogenetic tree window again.
- Select Sequence tree using ClustalW, and press the OK button. A phylogenetic tree based on ClustalW will be calculated and drawn as shown in Figure 5.7.26B.
- Quit VMD.

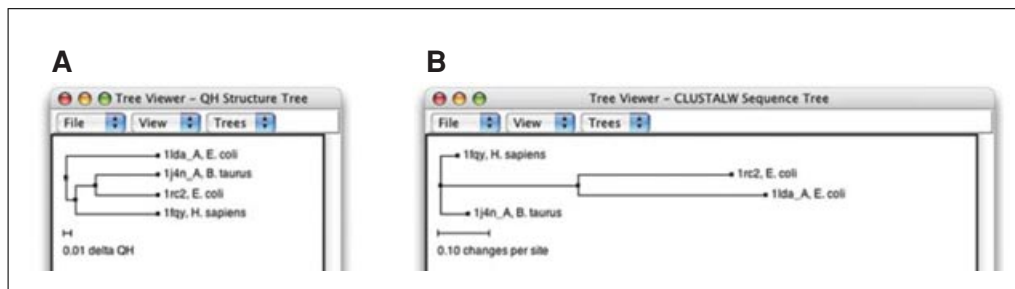


Figure 5.7.26 (A) A structure-based phylogenetic tree generated by Q_H values. (B) A sequence-based phylogenetic tree generated by ClustalW.

DATA ANALYSIS IN VMD

VMD is a powerful tool for analysis of structures and trajectories. Numerous tools for analysis are available under the VMD Main menu item Extension → Analysis. In addition to these built-in tools, VMD users often use custom-written scripts to analyze desired properties of the simulated systems. VMD Tcl scripting capabilities are very extensive, and provide boundless opportunities for analysis. In this section, we will learn how to use built-in VMD features for standard analysis, as well as consider a simple example of scripting.

Necessary Resources

Hardware

Computer

Software

VMD, a text editor, and a plotting application

Files

`ubiquitin.psf`, `pulling.dcd`, `equilibration.dcd` and `distance.tcl`, which can be downloaded at <http://www.currentprotocols.com>

Adding Labels in VMD

Labels can be placed in VMD to get information on a particular selection, to be used during visualization and quantitative analysis. Labels are selected with the mouse and can be accessed in Graphics → Labels menu. We will cover labels that can be placed on atoms and bonds, although angle and dihedral labelings are also available. In this context, labels for “bonds” or “angles” actually mean distances between two atoms or angles between three atoms, the atoms do not have to be physically connected by bonds in the molecule.

1. Start a new VMD session. Load the ubiquitin trajectory into VMD (using the files `ubiquitin.psf` and `pulling.dcd`). For graphical representation, display protein only, using NewCartoon for drawing method and Structure for coloring method. If you need help, see Basic Protocol 3, steps 1 to 6.
2. Choose the Mouse → Labels → Atoms menu item from the VMD Main menu. The mouse is now set to the mode for displaying atom labels. You can click on any atom on your molecule and a label will be placed for this atom. Clicking again on it will erase the label.
3. We will now try the same for bonds. Choose the Mouse → Label → Bonds menu item from the VMD Main menu. This selects the Display Label for Bond mode.

We will consider the distance between the α carbon of Lysine 48 and of the C terminus. In the pulling simulation, the former is kept fixed, and the latter is pulled at a constant force of 500 pN. In reality, polyubiquitin chains can be linked by a connection between the C terminus of one ubiquitin molecule and the Lysine 48 of the next. The simulation then mimics the effect of pulling on the C terminus with this kind of linkage.

4. Open the TkConsole window by selecting Extensions → Tk Console in the VMD Main menu. We will make a VDW representation for the α carbons of Lysine 48 and of the C terminus. To find out the indices of these atoms, make a selection including these two atoms by typing in the TkConsole window:

```
set sel [atomselect top ''resid 48 76 and name CA'']
```

BASIC PROTOCOL 14

Modeling Structure from Sequence

5.7.39

5. Get the indices by typing the following line in the TkConsole window:

```
$sel get index
```

This command should give the indices 770 1242.

Note that the atom numbers of these atoms in the pdb file are 771 and 1243. This is because VMD starts counting atom indices from zero. This is only the case for index, since VMD does not read them from the PDB file. Other keywords, such as residue, are consistent with the PDB file.

6. In the Graphical Representations window, create a representation for the selection `index 770 1242`, with VDW as drawing method.
7. Now that you can see the two α -carbons, choose the Mouse $\rightarrow$ Label $\rightarrow$ Bonds menu item from the VMD Main menu. Click on each atom one after the other.

You should get a line connecting the two atoms (Fig. 5.7.27). The number appearing next to the line is the distance between the two atoms in Ångströms. The value of the distance displayed corresponds to the current frame. Try playing the trajectory—you will see that the label is modified automatically as the distance between the atoms changes. Note that the appearance of the line (its color), as well as the appearance of essentially all other objects in VMD, can be changed in Graphics $\rightarrow$ Colors in the VMD Main menu.

The shortcut keys for labels are 1: Atoms and 2: Bonds. You can use these instead of the Mouse menu. Be sure the Open GL Display window is active when using these shortcuts.

8. The labels can be used not only for displaying, but also for obtaining quantitative information. In VMD Main menu, select Graphics $\rightarrow$ Labels. On the top left-hand side of the window, there is a pull-down menu where you can choose the type of label (Atoms, Bonds, Angles, and Dihedrals). For now, keep it in Atoms. You can see the list of atoms for which you have made a label.

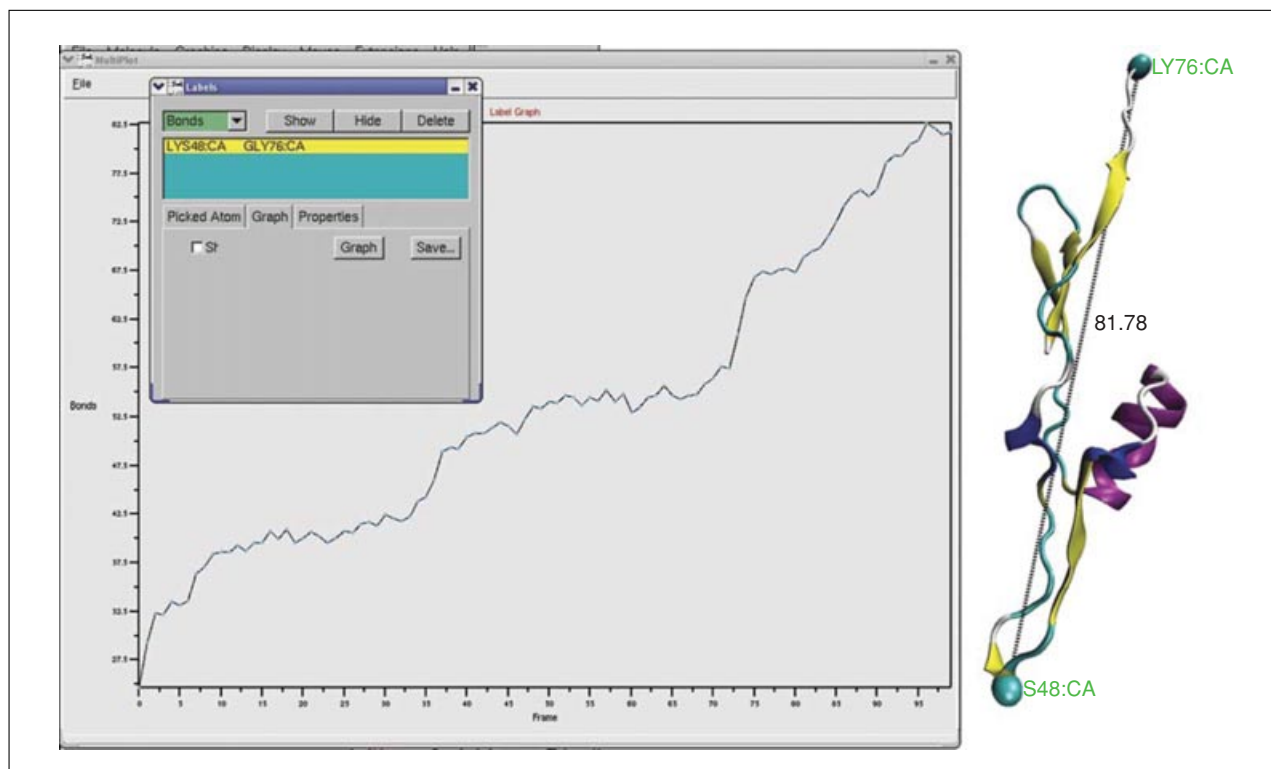


Figure 5.7.27 Labels in VMD. For the color version of this figure go to <http://www.currentprotocols.com>.

9. Click on one of the atoms. You can see all the information of the atom displayed on the bottom half of the Labels window. This information is useful to make selections; it corresponds to the current frame, and is updated as the frame is changed.
10. You can also delete, hide, or show the atom label by clicking on the corresponding button on the top of the Labels window.
11. In the Labels window, choose label type Bonds, and select the “bond” (distance) you labeled (Fig. 5.7.27). The information given corresponds to only the first atom in the bond, but the number in the Value field corresponds to the length of the bond in Ångstroms.
12. Click on the Graph tab. Select the bond you labeled between atoms 770 and 1242. Click on the Graph button. This will create a plot of the distance between these two atoms over time (Fig. 5.7.27). You can also save this data to a file by clicking on the Save button, and then use an external plotting program to visualize the data.
13. Quit VMD.

Example of a Built-In Analysis Tool: The RMSD Trajectory Tool

The built-in analysis tools in VMD are available under the menu item Extension → Analysis. These tools each feature a GUI window that allows one to enter parameters and customize the quantities analyzed. In addition, all tools can be invoked in a scripting mode, using the TkConsole window. We will learn how to work with one of the most frequently used tools, the RMSD Trajectory Tool.

In this example, we will analyze RMSD for two trajectories for the same system, `ubiquitin.psf`. One of them is the already familiar pulling trajectory, `pulling.dcd`, and the other is the trajectory of a simulation in which no force was applied to the protein, `equilibration.dcd`.

1. Start a new VMD session. Load the ubiquitin equilibration trajectory into VMD (using the files `ubiquitin.psf` and `equilibration.dcd`).
2. Choose Extension → Analysis → RMSD Trajectory Tool in the VMD Main window (Fig. 5.7.28). The RMSD Trajectory Tool window will show up.

In the RMSD Trajectory Tool window, you can see many customization options. For the default values, the molecule to be analyzed is `ubiquitin.psf` (the only one loaded). The selection for which RMSD will be computed is all of the protein atoms, excluding hydrogens (since the “noh” checkbox is on). The RMSD will be calculated for each frame with the reference to frame 0. Make sure the Plot checkbox is selected.

3. Click the Align button.

This will align each frame of the trajectory with respect to the reference frame (in this case, frame 0) to minimize the RMSD, by applying only rigid-body translations and rotations. This step is not necessary, but is desirable in most cases, because we are interested only in RMSD that arises from the fluctuations of the structure and not from the displacements and rotations of the molecule as a whole. The result of the alignment can be seen in the OpenGL display.

4. Click the RMSD button in the RMSD Trajectory Tool window. The protein RMSD (in Ångstrom) versus frame number is displayed in a plot (Fig. 5.7.28).

Over several initial frames, $RMSD = 0$ because positions of the protein atoms are fixed during that time in the simulation to allow water molecules around the protein to adjust to the protein surface. After that, the protein is released, and the RMSD grows quickly to around 1.5 Å. At that point, the RMSD levels off and remains at ~1.5 Å further on. This is a typical behavior for molecular dynamics simulations. Leveling of the RMSD means

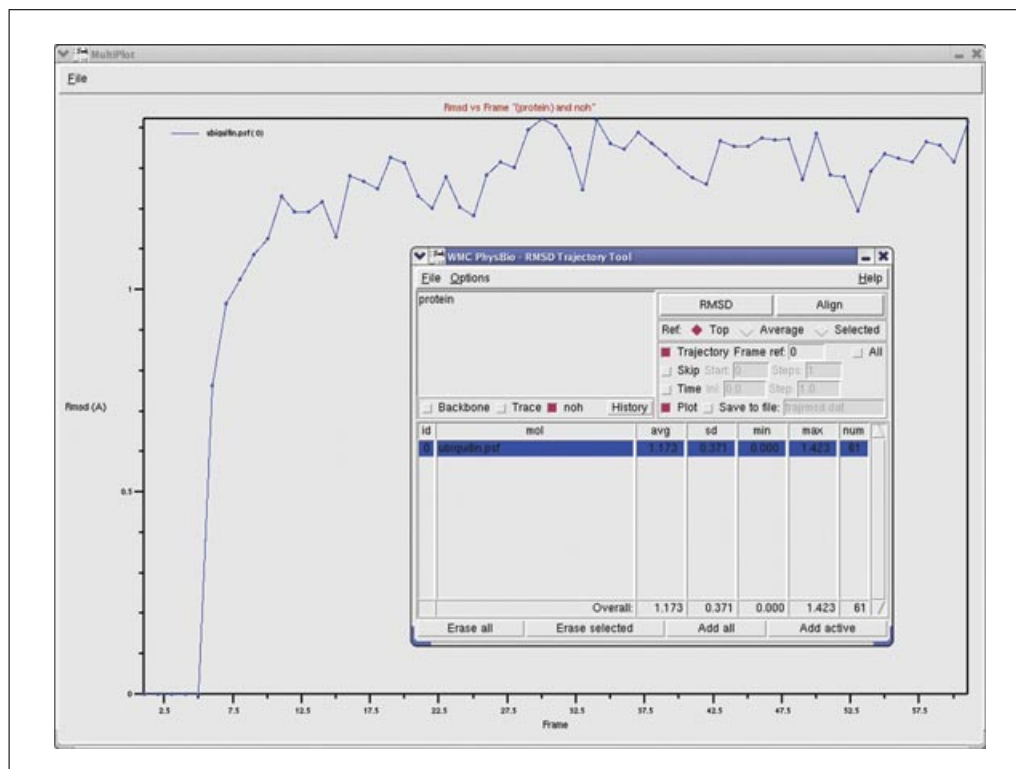


Figure 5.7.28 RMSD Trajectory Tool. The RMSD is plotted for the equilibration of ubiquitin.

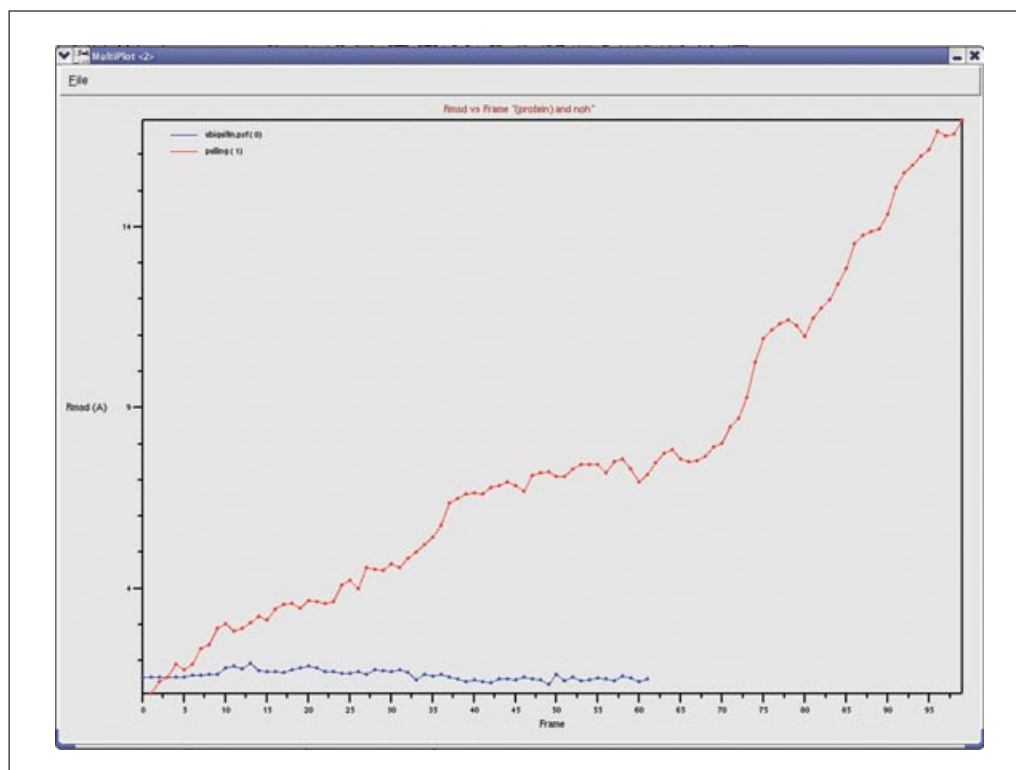


Figure 5.7.29 RMSD versus time for the equilibration (blue) and pulling (red) trajectories of ubiquitin. For the color version of this figure go to <http://www.currentprotocols.com>.

that the protein has relaxed from its initial crystal structure (which is affected by crystal packing and usually misses some atoms, e.g., hydrogens) to a more stable one. Production molecular dynamics simulations are usually preceded by such equilibration runs, where the protein is allowed to relax; the process is monitored by checking RMSD versus time, and equilibration is assumed to be sufficient when RMSD levels off. The RMSD of 1.5 Å is an acceptable value for most protein simulations. Usually, the deviations from the crystal structure in a simulation are due to the thermal motion and to the relaxation process mentioned; imperfections of the simulation force-fields contribute as well.

5. We will now work with the other trajectory, in which the ubiquitin is pulled apart. Load this trajectory into VMD using the files `ubiquitin.psf` and `pulling.dcd`. Make sure you load `ubiquitin.psf` as a new molecule. You can change the names of the molecules by double-clicking on them in the VMD Main menu (see Basic Protocol 9, steps 4 and 5).
6. In the RMSD Trajectory Tool window, hit the button Add all to update the list of molecules.
7. Click the Align button and then click RMSD button.

The new graph (Fig. 5.7.29) displays two RMSD plots versus time, one for the equilibration trajectory, and the other for the pulling trajectory. The RMSD for the pulling trajectory does not level off and is much higher than that in the equilibration trajectory, since the protein is stretched in the simulation.

8. Quit VMD.

Example of an Analysis Script

In many cases, one requires special types of trajectory analyses that are tailored for certain needs. The Tcl scripting in VMD provides opportunities for such custom tasks. Users commonly write their own scripts to analyze the features of interest. A very extensive library of VMD scripts, contributed by many users, is available online (http://www.ks.uiuc.edu/Research/vmd/script_library/). Here, we will explore a very simple exemplary script, `distance.tcl`, which computes the distance between two atom selections vs. time and the distribution of the distances.

1. Start a new VMD session. Load the ubiquitin equilibration trajectory (files `ubiquitin.psf` and `equilibration.dcd`).
2. Open the TkConsole window by selecting Extension → Tk Console in the VMD Main menu.
3. In the TkConsole window, load the script into VMD by typing: `source distance.tcl` (make sure that the file `distance.tcl` is in the current folder). This will load the procedure defined in `distance.tcl` into VMD.
4. One can now invoke the procedure by typing `distance` in the TkConsole window. In fact, the correct usage is

```
distance seltext1 seltext2 N_d f_r_out f_d_out
```

where `seltext1` and `seltext2` are the selection texts for the groups of atoms between which the distance is measured, `N_d` is the number of bins for the distribution, and `f_r_out` and `f_d_out` are the file names to where the output distance versus time and distance distribution will be written.

5. Open the script file `distance.tcl` with a text editor. You can see that the script does the following:

BASIC PROTOCOL 16

Modeling Structure from Sequence

5.7.43

- a. Choose atom selections:


```
set sel1 [atomselect top '$seltext1']
set sel2 [atomselect top '$seltext2']
```
- b. Get the number of frames in the trajectory and assign this value to the variable `nf`:


```
set nf [molinfo top get numframes]
```
- c. Open file specified by the variable `f_r_out`:


```
set outfile [open $f_r_out w]
```
- d. Loop over all frames


```
for {set i 0} {$i < $nf} {incr i} {
```
- e. Write out the frame number and update the selections to the current frame:


```
puts 'frame $i of $nf'
$sel1 frame $i
$sel2 frame $i
```
- f. Find the center of mass for each selection (`com1` and `com2` are position vectors):


```
set com1 [measure center $sel1 weight mass]
set com2 [measure center $sel2 weight mass]
```
- g. At each frame `i`, find the distance by subtracting one vector from the other (command `vecsub`) and computing the length of the resulting vector (command `veclength`), assign that value to an array element `simdata($i.r)`, and print a frame-distance entry to a file:


```
set simdata($i.r) [veclength [vecsub $com1 $com2]]
puts $outfile '$i $simdata($i.r)'
}
```
- h. Close the file:


```
close $outfile
```
- i. The second part of the script is for obtaining the distance distribution. It starts from finding the maximum and minimum values of the distance.


```
set r_min $simdata(0.r)
set r_max $simdata(0.r)
for {set i 0} {$i < $nf} {incr i} {
  set r_tmp $simdata($i.r)
  if {$r_tmp < $r_min} {set r_min $r_tmp}
  if {$r_tmp > $r_max} {set r_max $r_tmp}
}
```
- j. The step over the range of distances is chosen based on the number of bins `N_d` defined in the beginning and all values for the elements of the distribution array are set to zero.


```
set dr [expr ($r_max - $r_min) / ($N_d - 1)]
for {set k 0} {$k < $N_d} {incr k} {
  set distribution($k) 0
}
```
- k. The distribution is obtained by adding 1 (`incr ...`) to an array element every time the distance is within the respective bin:


```
for {set i 0} {$i < $nf} {incr i} {
  set k [expr int(($simdata($i.r) - $r_min) / $dr)]
  incr distribution($k)
}
```

1. Write out the file with the distribution:

```
set outfile [open $f_d.out w]
for {set k 0} {$k < $N_d} {incr k} {
  puts $outfile "[expr $r_min + $k* $dr]\n"
  $distribution($k)
}
close $outfile
```

6. Now, run the script by typing in the TkConsole window:

```
distance ''protein'' ''protein and resid 76'' 10\
res76-r.dat res76-d.dat
```

This will compute the distance between the center of the protein and center of the terminal residue 76, and write the distance versus time and its distribution to files `res76-r.dat` and `res76-d.dat`.

7. Repeat the same for the protein's residue 10 by typing in the TkConsole window:

```
distance ''protein'' ''protein and resid 10'' 10\
res10-r.dat res10-d.dat
```

The data in files produced by the script `distance.tcl` are in two-column format. Compare the outputs for residue 76 and 10 using your favorite external plotting program (Fig. 5.7.30).

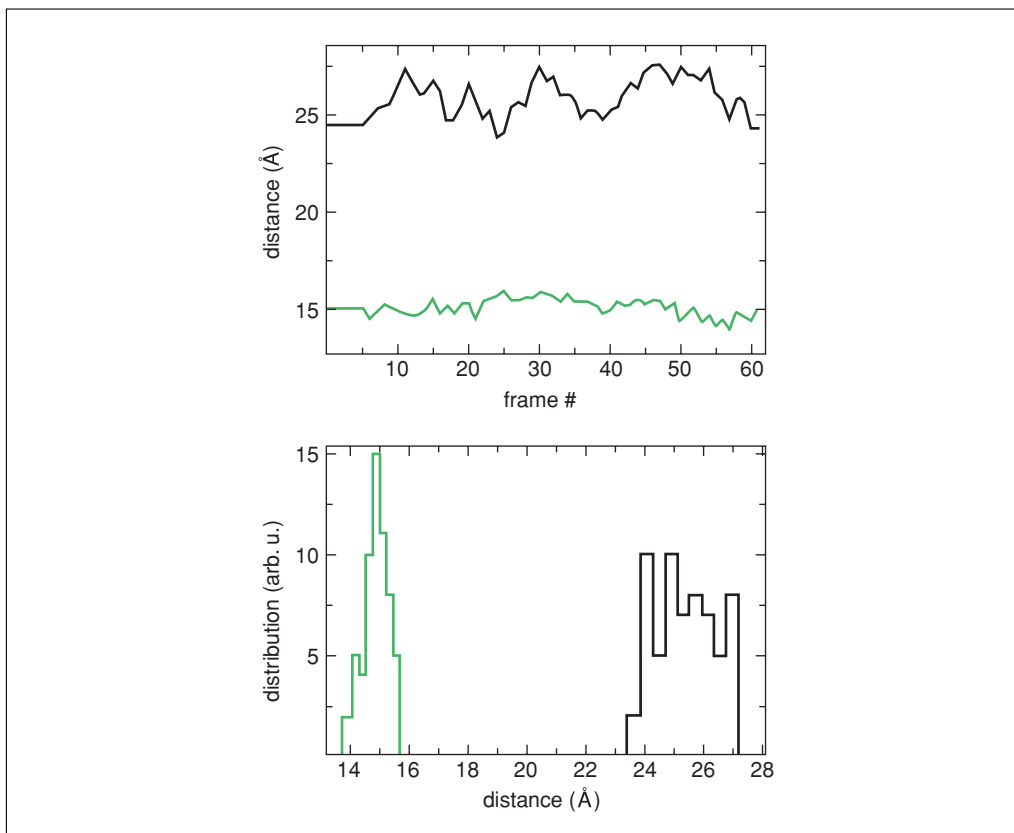


Figure 5.7.30 Distance between a residue and the center of ubiquitin. The distances analyzed are those for residue 76 (black) and residue 10 (green). For the color version of this figure go to <http://www.currentprotocols.com>.

Residue 76 is at the protein's C-terminus, which is extended towards the solvent and is quite flexible, while residue 10 is at the surface of the globular part of ubiquitin. The difference in their dynamics with respect to the rest of the protein is immediately obvious when our newly obtained data are plotted (Fig. 5.7.30): the distance of residue 76 from the protein's center is substantially greater than that of residue 10, and the distribution of the distance is noticeably wider due to the flexibility of the C-terminus. This is just a simple example of scripting for the analysis of a trajectory. Similar, but usually much more complex, customized scripts are routinely employed by VMD users to perform many kinds of analysis.

8. Quit VMD.

COMMENTARY

Background Information

VMD has been developed by the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. Throughout its development, many features have been added, and user-specific functions can be implemented through embedded scripting languages like Python and Tcl, providing a wide spectrum of tools for the scientific community. Specifically, VMD is most suitable for high-resolution visualization and image rendering, preparation of molecular dynamics simulation systems and analysis of simulation results, and animation of molecular dynamics trajectory. In addition, VMD can also work with volumetric data, and provides a platform for bioinformatics analysis such as protein sequence alignment. What we are able to present in this tutorial only showcases a small part of VMD's capability. But now that you have learned the basics of VMD, you are ready to explore its many other features most suitable for your research. For this purpose, there are many tutorials available that aim at offering a more focused training, either on a specific tool or on a scientific topic. You can find many useful documentations, including the comprehensive VMD User's Guide, at the VMD homepage <http://www.ks.uiuc.edu/Research/vmd/>.

Critical Parameters and Troubleshooting

Most parameters in VMD can be easily adjusted to suit individual users' needs. For example, when rendering molecules using a representation, as described in Basic Protocol 1, users can adjust the resolution of the representation in the graphical user interface, as well as many other parameters specific to the drawing method of the representation. New users of VMD might find default settings for most parameters are good starting points, but are also encouraged to change the parameters and test the difference. If you have any ques-

tions on using VMD, we encourage you to subscribe to the VMD mailing list http://www.ks.uiuc.edu/Research/vmd/mailling_list/.

Acknowledgments

This tutorial is largely based on the following VMD tutorials, case studies, and user's guides. We hence would like to thank these authors who have provided this tutorial its starting form:

Jordi Cohen, Marcos Sotomayor, and Elizabeth Villa, "VMD Molecular Graphics."

Alek Aksimentiev, John Stone, David Wells, and Marcos Sotomayor, "VMD Images and Movies Tutorial."

Fatemeh Khalili, Elizabeth Villa, Yi Wang, Emad Tajkhorshid, Brijeeet Dhaliwal, Zan Luthey-Schulten, John Stone, Dan Wright, and John Eargle, "Aquaporins with the VMD MultiSeq Tool."

VMD has been developed by the Theoretical and Computational Biophysics Group at the University of Illinois and the Beckman Institute, and is supported by funds from the National Institutes of Health and the National Science Foundation.

Citing VMD

The development of VMD is funded by the National Institute of Health. Proper citation is a primary way in which we demonstrate the value of our software to the scientific community, and is essential to continued NIH funding for VMD. The authors request that all published work, that utilizes VMD include the primary VMD citation at a minimum:

Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics," J. Molec. Graphics, 1996, vol. 14, pp. 33-38.

Work that uses softwares or plugins incorporated into VMD should also add the proper citations for those tools. For example, work that uses MultiSeq as introduced in Basic Protocols 11 to 13 should cite:

Roberts, E., Eargle, J., Wright, D. and Luthey-Schulten Z., "MultiSeq: Unifying sequence and structure data for evolutionary analysis," *BMC Bioinformatics*, 2006, 7:382.

Please see <http://www.ks.uiuc.edu/Research/vmd/allversions/cite.html> for more information on how to cite VMD and its tools.

Literature Cited

Cruz-Chu, E.R., Aksimentiev, A., and Schulten, K. 2006. Water-silica force field for simulating nanodevices. *J. Phy. Chem. B* 110:21497-21508.

Eastwood, M.P., Hardin, C., Luthey-Schulten, Z., and Wolynes, P.G. 2001. Evaluating protein structure-prediction schemes using energy landscape theory. *IBM J. Res. Dev.* 45:475-497.

Freddolino, P.L., Arkhipov, A.S., Larson, S.B., McPherson, A., and Schulten, K. 2006. Molecular dynamics simulations of the complete satellite tobacco mosaic virus. *Structure* 14:437-449.

Frishman, D. and Argos, P. 1995. Knowledge-based secondary structure assignment. *Proteins* 23:566-579.

Humphrey, W., Dalke, A., and Schulten, K. 1996. VMD—Visual Molecular Dynamics. *J. Mol. Grap.* 14:33-38.

Israelowitz, B., Gao, M., and Schulten, K. 2001. Steered molecular dynamics and mechanical functions of proteins. *Curr. Opin. Struct. Biol.* 11:224-230.

Murata, K., Mitsuoka, K., Hirai, T., Walz, T., Agre, P., Heymann, J.B., Engel, A., and Fujiyoshi, Y. 2000. Structural determinants of water permeation through aquaporin-1. *Nature* 407:599-605.

Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L., and Schulten, K. 2005. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* 26:1781-1802.

Roberts, E., Eargle, J., Wright, D., and Luthey-Schulten, Z. 2006. MultiSeq: Unifying sequence and structure data for evolutionary analysis. *BMC Bioinformatics*. 7:382.

Russell, R.B. and Barton, G.J. 1992. Multiple protein sequence alignment from tertiary structure comparison: Assignment of global and residue confidence levels. *Proteins* 14:309-323.

Savage, D.F., Egea, P.F., Robles-Colmenares, Y., O'Connell, J.D. III, and Stroud, R.M. 2003. Architecture and selectivity in aquaporins: 2.5 Å X-ray structure of aquaporin Z. *PLoS Biol.* 1:E72.

Sotomayor, M., Vasquez, V., Perozo, E., and Schulten, K. 2007. Ion conduction through MscS as determined by electrophysiology and simulation. *Biophys. J.* 92:886-902.

Sui, H., Han, B.-G., Lee, J.K., Walian, P., and Jap, B.K. 2001. Structural basis of water-specific transport through the AQP1 water channel. *Nature* 414:872-878.

Tajkhorshid, E., Nollert, P., Jensen, M.Ø., Miercke, L.J.W., O'Connell, J., Stroud, R.M., and Schulten, K. 2002. Control of the selectivity of the aquaporin water channel family by global orientational tuning. *Science* 296:525-530.

Thompson, J.D., Higgins, D.G., and Gibson, T.J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* 22:4673-4680.

Törnroth-Horsefield, S., Wang, Y., Hedfalk, K., Johanson, U., Karlsson, M., Tajkhorshid, E., Neutze, R., and Kjellbom, P. 2006. Structural mechanism of plant aquaporin gating. *Nature* 439:688-694.

Vijay-Kumar, S., Bugg, C.E., and Cook, W.J. 1987. Structure of ubiquitin at 1.8Å resolution. *J. Mol. Biol.* 194:531-544.

Wang, Y., Cohen, J., Boron, W.F., Schulten, K., and Tajkhorshid, E. 2007. Exploring gas permeability of cellular membranes and membrane channels with molecular dynamics. *J. Struct. Biol.* 157:534-544.

Yin, Y., Jensen, M.Ø., Tajkhorshid, E., and Schulten, K. 2006. Sugar binding and protein conformational changes in lactose permease. *Biophys. J.* 91:3972-3985.

Yu, J., Yool, A.J., Schulten, K., and Tajkhorshid, E. 2006. Mechanism of gating and ion conductivity of a possible tetrameric pore in Aquaporin-1. *Structure* 14:1411-1423.

Supplemental Files

Supplemental files can be downloaded from <http://www.currentprotocols.com> by clicking "Current Protocols" beneath the Bioinformatics head and following the Sample Datasets link.

1fqy.pdb

pdb coordinate file for human aquaporin (Murata et al., 2000)

1j4n.pdb

pdb coordinate file for bovine aquaporin (Sui et al., 2001)

1lda.pdb

pdb coordinate file for E. coli GlpF (Tajkhorshid et al., 2002)

1rc2.pdb

pdb coordinate file for E. coli aquaporin (Savage et al., 2003)

1ubq.pdb

pdb coordinate file for ubiquitin (Vijay-Kumar et al., 1987)

beta.tcl

An example tcl script.

distance.tcl

An example tcl script.

equilibration.dcd

dcd molecular dynamics trajectory file of an equilibration simulation

`pulling.dcd`

dcd molecular dynamics trajectory file of a protein-pulling simulation

`spinach_aqp.fasta`

An example fasta protein sequence file.

`ubiquitin.psf`

psf structure file for ubiquitin that defines connectivity of atoms